(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 821 817 B1

(12) # EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
23.06.1999 Bulletin 1999/25

(21) Application number: 96905192.9

(22) Date of filing: 17.01.1996

(51) Int Cl.^{6}: G06F 19/00, G06F 9/44

(86) International application number:
PCT/US96/00883

(87) International publication number:
WO 96/22575 (25.07.1996 Gazette 1996/34)

(54) **CONTROL SYSTEMS BASED ON SIMULATED VIRTUAL MODELS**
AUF SIMULIERTEN VIRTUELLEN MODELLEN BASIERENDES STEUERUNGSSYSTEM
SYSTEMES DE COMMANDE BASES SUR DES MODELES VIRTUELS SIMULES

(84) Designated Contracting States:
CH DE GB LI

(30) Priority: 17.01.1995 US 373688
17.01.1995 US 373992

(43) Date of publication of application:
04.02.1998 Bulletin 1998/06

(73) Proprietor: INTERTECH VENTURES, LTD.
Wellesley, MA 02181 (US)

(72) Inventor: THALHAMMER-REYERO, Cristina
Framingham, MA 01701 (US)

(74) Representative:
Wildhack, Helmut, Dipl.-Ing. Dr. et al
Patentanwälte Dipl.-Ing. Leo Brauneiss,
Dipl.-Ing. Dr. Helmut Wildhack,
Dipl.-Ing. Dr. Gerhard Jellinek,
Landstrasser Hauptstrasse 50
1030 Wien (AT)

(56) References cited:
EP-A- 0 367 544

• ELEKTRONIK, vol. 40, no. 22, 29 October 1991, pages 122-123, 128 - 134, XP000268407 KIEZULAS C J ET AL: "ENTWICKLUNG VON EXPERTENSYSTEMEN"

Description

I. TECHNICAL FIELD

[0001] The present invention in its broadest form relates to computer-based systems, methods and visual interfaces for providing an integrated development and deployment framework for visual modeling and dynamic simulation of Virtual Models of complex systems, which can be further integrated with monitoring and control devices to control the operation of the complex systems modeled and can be used for information retrieval.

Copyright Notice

[0002] A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

II. BACKGROUND ART

A. Knowledge-Based and Model-Based System for Monitoring and Control

[0003] Process industries, including the pharmaceutical, biotechnology, chemical, food, environmental and others may use artificial intelligence for process optimization to control complex productions facilities. Current systems monitor very general types of phenomena, such as gas pressure, pH, and in some occasions, the concentration of some product that correlates with cell growth or production, but those parameters are usually poor indicators of how much of the desired product is produced. Other methods for designing monitoring and control systems for laboratory and industrial applications have been described, such as the one described in the patent application published as EP 0 367 544 A2 (Int. Dev. Res. Center) 9 May 1990, which uses a graphical interface to graphically model the set of instruments and controllers of such monitoring and control systems, and and a natural language to allow the integration of the knowledge of experts into the automated control facilities,. Most monitoring systems are concerned with the overall processes that occur within the physical contstraints of given reactions tanks, but do not model the many compartmentilized subsystems contained in each of those tanks with biological systems, to more finely tune the productivity of those subsystems. **Object-oriented expert systems** allow a powerful knowledge representation of physical entities and conceptual entities. Each class defines each of the attributes characteristic of that class and distinguish it from another types of objects. Manipulation and retrieval of the values of the data structures may be performed through methods attached to a object's class. **Model-based systems** can be derived from empirical models based on regression of data or from first-principle relationships between the variables.

[0004] There is a number of **commercially available shells** and toolkits that facilitate the development and deployment of domain-specific knowledge-based applications. Of those, real-time expert-system shells offer capabilities for reasoning on the behavior of data over time. Each of the shells from various vendors offers its set of advantages and offers a different level of graphic sophistication. The shell selected for the implementation of this invention is Gensym Corporation's G2, which is designed for complex and large on-line applications where large number of variables can be monitored concurrently. It is able to reason about time, to execute both time-triggered and event-triggered actions and invocations, to combine heuristic and procedural reasoning, dynamic simulation, database interfaces, and other facilities that allow the knowledge engineer to concentrate on the representation and incorporation of domain-specific knowledge to create domain-specific applications. G2 provides a built-in inference engine, a simulator, prebuilt libraries of functions and actions, developer and user-interfaces, and the management of their seamless interrelations. A built-in inspect facility permits users to search for, locate, and edit various types of knowledge. Task prioritization, asynchronous concurrent operations, and real-time task scheduling are automatically provided by this shell. G2 also provides a graphic user interface builder, which may be used to create graphic user interfaces which are language independent and allow to display information using colors, pictures and animation. Dynamic meters, graphs, and charts can be defined for interactive follow-up of the simulation. It also has debugger, inspect and describe facilities. The knowledge bases can be saved as separated modules. The graphic views can be shared with networked remote CPUs or terminals equipped with X Windows server software.

B. Computer-Aided Physiological Modeling and Artificial Intelligence in Molecular Biology

[0005] Most computer-aided physiological and molecular modeling approaches have resulted in computer models of physiological function that are numerical mathematical models that relate the physiological variables using empirically determined parameters. Those models, which can become quite complex, aim at modeling the overall system.

[0006] Both molecular biology and medicine have been fields of previous activity in the application of artificial intelligence (AI). In molecular biology, although there were some early systems such as Molgen and Dendral, the activity has intensified recently as a consequence of the explosion in new technologies and the derived data, mostly related with the Human Genome project and the handling of large amounts of sequence data generated, relating to both DNA and proteins. For a current state of the art, see the topics covered in symposia such as the recent Second International Conference on Intelligent Systems for Molecular Biology, 1994, Stanford University, CA. (its Proceedings are here included by reference). Here, only two projects will be mentioned that have some common objectives with the system that is the object of this invention. Discussions over other previous approaches are also included in those references.

[0007] The Molgen group at Stanford University has studied scientific theory formation in the domain of molecular biology, as reported by Karp, P.D.and Friedland. P. (included here by reference). This project relates to the system of this invention in that both "are concerned with biochemical systems containing populations of interacting molecules ... in which the form of knowledge available ... varies widely in precision from quantitative to qualitative", as those authors write. The Ph.D. dissertation of P. D. Karp (included here by reference) "developed a qualitative biochemistry for representing theories of molecular biology", as he summarizes in an abstract in AI Magazine, Winter 1990, pp 9-10. He developed three representation models to deal with biochemical pathways, each having different capabilities and using different reasoning approaches. Model 1 uses IntelliCorp's KEE frames to describe biological objects and KEE rules to describe chemical reactions between the objects, which he recognizes to have serious limitations because is not able to represent much of the knowledge available to biologists. The objective of Model 2 is to predict reaction rates in a given reaction network, incorporating a combination of quantitative and qualitative reasoning about state-variables and their interdependencies. The drawbacks are that this model is not able to incorporate a description of the biological objects that participate in the reactions, and it does not have temporal reasoning capabilities, representing just an static description of the state variables and their relationships. The third model, called GENSIM and used for both prediction and hypothesis formation, is an extension of Model 1 and is composed of three knowledge bases or taxonomical hierarchies of classes of a) biological objects that participate in a gene-regulation system, b) descriptions of the biological reactions that can occur between those objects, and c) experiments with instances of those classes of objects. The GENSIM program predicts experimental outcomes by determining which reactions occur between the objects in one experiment, that create new objects that cause new reactions. Characteristics of the GENSIM program that may be relevant, although different, for the system of this invention are: a) chemical objects are homogeneous populations of molecules, objects can be decomposed into their component parts, and identical objects synthesized during a simulation are merged; b) chemical processes represent reactions between those populations as probabilistic events with two subpopulations, one that participate in the reaction and one that does not. Those processes can create objects and manipulate their properties, but cannot reason about quantitative state variables such as quantities. In his words, "processes ... specify actions that will be taken if certain conditions hold", and in that sense are like production-rules; c) restrictions are specified in the form of preconditions for chemical reactions to happen; and d) temporal reasoning is not available, resulting again in static representations and simulating only behavior in very short time intervals.

[0008] The system of this invention integrates a variety of forms of knowledge representation, some of them totally novel, while some of these forms may have been treated by other authors similarly in some aspects. However, upon integration into a totally new approach, that treatment becomes a part of a novel representation and innovative system. Regarding the semi-quantitative simulation component this invention, LE. Widman (1991) describes a semi-quantitative simulation of dynamic systems in a different domain, with the assumption that"... questions can be answered in terms of relative quantities rather than absolute quantities ... model parameters that are not specified explicitly are given the implicit, default values of 'normal' (unity) ...". As it will become clear from the detail descriptions in the following sections, the innovative tools and methods used in the present implementation are quite different. For example, while he assumes that" ... the default, or implicit, value of 'normal' maps onto unity for parameters and onto zero for variables...." the assumption in the prebuilt modular components in the current implementation differs in that the default value of 'normal' may map onto values other than unity and zero, with those values being defined based on expert knowledge.

III. DISCLOSURE OF INVENTION

[0009] This invention describes an integrated computer-based system, methods and visual interfaces for providing a development and deployment framework for visual modeling and dynamic simulation of Virtual Models of complex systems, which can be further integrated with monitoring and control devices to monitor and control the operation of the complex systems modeled, based on the real-time simulation of those Virtual Models. The system of this invention can be used by scientists to build the Virtual Models, which in turn are to be used by scientists or process engineers to design, monitor and control a variety of processing units. The Virtual Models can also be used for information retrieval, by using the set of visual interfaces provided to perform a variety of tasks. The available information and data about those complex models is stored into modular, modifiable, expandable and reusable knowledge structures, with several layers of encapsulation which allows to hide or display the details at the desired level of complexity. More particularly, the Virtual Models in the present invention describe visual models of biochemical complex systems, comprising sets of icons representing processes and their participants linked into multidimensional pathways, further organized in a hierarchy of icons representing discrete time and space compartments, wherein such compartments may contain other compartments, and wherein those modular icons encapsulate in different layers all the information, data, and mathematical models that characterize and define each Virtual Model.

[0010] Some process industries use large-scale cultivation of microorganisms or mammalian cells, which are extreme cases in terms of complexity when considering those cells as the individual manufacturing plants involved in complex chemical synthesis. Microorganisms are the preferable systems for producing natural substances that have a multitude of uses, such as drugs. foods, additives, biodetergents, biopolymers, and other new and raw materials. Mammalian cells are the preferable systems for producing potent active substances for therapeutic and diagnostic uses. The production of a particular secreted protein that is produced in very low amounts in relation to other proteins could be finely controlled externally by modifying the types and amounts of inputs added, if one could predict what will happen by adding those inputs, which requires a good knowledge and a model of such system of reactions. This is particularly the case with biological cellular systems that have a very sophisticated methods to transduce the signals provided by ligands in their external environment to the interior of the cell, resulting in the execution of specific functions. Such detailed and accessible mechanistic models of those pathways of reactions are not currently used for monitoring and control systems, but would be highly desirable. This invention provides the system and methods that allows scientists to visually build detailed mechanistic models of the complex systems involved, and to further develop and use inference methods to integrate the simulation of those Virtual Models with inputs from monitoring devices, to allow for the intelligent control of the operation of the complex system.

[0011] The accuracy and validity of knowledge-based systems correlates not only with the quality of the knowledge available to the developers but also with their ability to understand, interpret and represent that knowledge. Because of the complex interrelationships driving the biochemical processes within and between cells, it is necessary to provide the many options for knowledge representation required by those systems. This invention presents a hybrid dynamic expert system that combines: a) the inheritance and encapsulation features characteristic of an object-oriented modeling approach, b) the procedural and rule-based inference capability of an expert system, and c) a model-based simulation capability. It is an objective of this invention to provide a framework that: a) allows domain-experts to directly enter their knowledge to create visual models, and to modify them as needed based on additional experimentation, without the need of intermediaries, and b) provides a knowledge-base having a well defined structure for representing knowledge about entities, populations of entities, processes, pathways and interacting networks of pathways, providing a visual interface and associated methods capable of dealing with incomplete and constantly evolving information and data. The data structures and domain-specific knowledge-base are independent of application-specific use, allowing the application-specific knowledge-bases to expand without affecting the basic operation of the system. Specific applications can be quickly built by using the prebuilt building blocks and the paradigm of "Clone, Link, Configure. and Initialize". The innovations of this invention include but are not limited to the specific design, integration, and use of libraries of building-blocksfor representation, interpretation, modeling and simulation of different types of entities and their states, their relations and interactions, the pools of each of those entities in different compartments, the processes in which those pools of entities participate, and their discrete compartmentalization in time and space. A large Virtual Model can be built as a set of modules, focusing each on different subsystems. Each of the modules can be run on top of the repository module that contain the class definitions and associated methods, they can be dependent or independent of each other, or they can be maintained in separate CPUs and seamless integrated by the Shell's supervisor. Such libraries provide the prebuilt building-blocks necessary for visually representing the vast breath of knowledge required for building Virtual Models of complex systems in general, and of biological systems in particular. The building-blocks are classified in palettes that can be selected through system menus. The descriptive information, data, and the mathematical models are all encapsulated within the modular components, in the form of attributes or in the form of component icons, with a plurality of methods associated with each of the icons. Of importance in simulating the behavior of complex systems is the need to model the different quantities and states at which the entities can be found in particular locations at different points in time, and also to model the events that cause the transitions from one state to another, or the translocation from one location to another, or the progress to the next phase in the time sequence. Teachings of this invention comprise: the representation of those states, transitions, and locations; and the methods to dynamically simulate the dynamics of the pools of entities in each state, location, or phase. In the particular domain implemented to illustrate this invention, there are several major types of states and transitions to be considered, depending of whether the entity to be considered is a biological system, organ, cell, cellular compartment, molecule or any other of their components. F. The system of this invention allows various bioProcesses to simultaneously compete for the contents of a bioPool while also allows one bioPool to participate in various bioProcesses in different capacities, with different units within a bioPool behaving in different ways. The result is a very complex multidimensional network of composite bioObjects, which provides the matrix for further reasoning and simulation by the program The formulas that provide the values for the encapsulated sets of variables refer to linked bioObjects,, providing the capability of concurrently and dynamically compute a large number of processors arranged both in parallel and in series within that matrix. Such set of model differential and algebraic equations, together with the set of associated parameters, control the behavior of the variables and the system as a whole. The system's variables and parameters are embedded and distributed throughout the system of connected structures, encapsulated within the subcomponents that define the system's architecture. The compartmentalized bioModels, and the methods attached to them, encode knowledge that enables the program to reason about the containment of different parts of the model in several compartments, while the architecture of the network of linked bioObjects of diverse types is transparently maintained, regardless of the transfer of the bioObject icons to different locations. The modeler can define expert rules to monitor the values of any of those variables while the simulation is running, and to either set other values or control the course of the simulation in a variety of ways. The expert rules can also reason about time or about events resulting from the simulation, and direct further actions to be executed by the system.

[0012] The visual interface further provides quick access to several automated methods for compiling, retrieving, and displaying the modular components of the visual models as well as the information and data they contain. A number of functions and visual interfaces can be selected from the menus associated with each of those icons or their components, to extract in various forms the information contained in the models build with those building blocks, such as:

a) methods to create and display interactive networks of pathways, by programmatically integrating the components of the Virtual Model, including branching and merging of pathways, cross-talk between pathways that share elements, and feed-back and forward loops;
b methods to perform complex predefined queries that combine criteria related to the structural composition of the bioEntities involved, the position of bioPools downstream or upstream of the bioPool taken as reference, the role that those poolsof entities play in processes, the location of those pools and processes in the discrete time and space compartments, or any combination of them.
c methods to dynamically simulate their continuous interactions, modifications, and translocations to other compartments, and the time-dependency of such interactions, optionally using the encapsulated absolute-valued or scaled-valued parameters and variables.

[0013] The examples provided to document the current implementation of this invention focus on modeling biochemical regulatory processes that are relevant for intracellular or intercellular signaling. This is however one of the many potential applications of the system of this invention, and it should not be construed to limit the applicability of this system to the numerous other applications that involve complex systems in any other domain, which can be developed by minor modifications or additions of the system using the methodology here presented. There are numerous other uses of the core system, methods and visual interface of this invention, in addition to the monitoring and control applications described here. Of particular interest is the modeling and simulation of disease specific conditions and the testing the effects - both desired and unwanted side effects- of potential therapeutic agents. This invention also allows to analyze disturbances such as potential environmental or biological inducers of disease in both physiological and pathophysiological models. The methods included in this invention can in a similar way be used in a variety of applications, including but not limited to: simulation and prediction of experimental results; interactive drug design; study of drug side-effects and multiple drug interactions: simulation of interactive causes in the induction and progression of disease, including both biological and environmental factors; diagnostics and clinical decision support; therapy planning; theoretical research; and numerous other applications.

[0014] The foregoing and additional objectives, descriptions, features, operations and advantages of the present invention will be understood from the following detailed description of the preferred embodiments in combination with the accompanying figures.

IV. BRIEF DESCRIPTION OF DRAWINGS

[0015] FIG.1 is a high-level illustration of the various components integrated in the system of this invention to use the Virtual Models for control functions.

[0016] FIG.2 is an schematic representation of the organization in the system of this invention of the components of the Virtual Models in the domain of cell biology.

[0017] FIG.3 is an schematic representation of the organization in of domain-specific processes in discrete space compartments an time compartments.

[0018] FIG.4 is an schematic representation of the handling of the dynamics of the progression of populations of cells through different states by means of the sets of pools of cells and processes characteristic of this invention.

[0019] FIG.5 is an schematic representation of the multiple layers of linked pools of entities and processes the result in the multidimensional pathways characteristic of this invention.

[0020] FIG.7 is a detailed representation focusing on the iconic components of a process and their relations to its sources of inputs and the targets for its outputs.
[0021] FIG.8 is a detailed representation focusing on the iconic components of a pool of entities and its inputs and outputs.
[0022] FIG.12 describes the sets of attributes of the components of a reservoir, focusing on the variables and parameters of its model-block component..
[0023] FIG.13 describes the sets of attributes of the components of a pool of entities, focusing on its variables and parameters.
[0024] FIG.14 describes the sets of attributes of the components of a process, focusing on the variables and parameters of its reactants.
[0025] FIG.15 describes the sets of attributes of the components of a process, focusing on the variables and parameters of its engine and products.
[0026] FIG.16 shows domain menus of the modeler mode that allow modelers to access the different palettes of prebuilt building-blocks.
[0027] FIG.18 shows an example of a complex molecular structure built with the prebuilt molecular components.
[0028] FIG.19 shows a palette with examples of prebuilt model-blocks.
[0029] FIG.20 shows a palette with examples of prebuilt composite processes.
[0030] FIG.22 describes the tools for interactively establishing links between components that result in multidimensional pathways.
[0031] FIG.24 describes a domain-specific compartmentalization of the components of a Virtual Model, in this case focusing on the sequential phases of a cell's cycle and on its subcellular compartments.
[0032] FIG.25 describes a domain-specific implementation of compartmentalized model interacting with each other and with the external environment.
[0033] FIG.26 describes a domain-specific characterization of the compartmentalized model from an external point of view.
[0034] FIG.28 is an schematic representation of the combination of the inference and simulation capabilities used in this invention to simulate Virtual Models of complex systems.
[0035] . FIG.29 is a detailed representation of how in this invention pools of cells interact with pools of molecules as reactants of processes which products are either molecules or cells in a different state.
[0036] FIG.30 describes an example of the predefined Query Panels and their use.
[0037] FIG.31 describes how a user can request from any process within the Virtual Model the dynamic generation of the pathways of all the processes that are either upstream or downstream fro m that process.
[0038] FIG.32 describes an example of the predefined Navigation Panels that the user can request from any reservoir within the Virtual Model for the dynamic generation of constrained pathways of all the processes that are either upstream or downstream fro m that reservoir, but which are contained within a compartment selected by the user.
[0039] FIG.33 describes an example of the predefined Simulation Panels that the user can request from any reservoir within the Virtual Model the dynamic generation of constrained pathways and for the control of the dynamic simulation of the kinetics of those pathways.
[0040] FIG.34 describes how such simulation can be followed by plotting the time-series of any of the quantitative values of any reservoir and process in the subsystem being simulated.
[0041] FIG.35 describes an example of the predefined Experiment Panels that the user can select from the domain menus to request the dynamic generation of constrained pathways from multiple initial points and for the control of the dynamic simulation of the kinetics of those pathways.
[0042] FIG.36 describes an example of such set of pathways from multiple initial points .

V. BEST MODE FOR CARRYING OUT THE INVENTION

Notes

[0043] The body of the present application has sections that may contain some discussion of prior art teachings, intermingled with of innovative and specific discussion of the best mode to use that prior art in this invention as presently contemplated. To describe the preferred embodiments, it is necessary to include in the discussion the capabilities offered by the shell used as development and deployment framework for this invention (hereafter referred to as "the Shell"). The applicant specifically notes that statements made in any of those sections do not necessarily delimit the various inventions claimed in the present application, but rather are included to explain how the workings of an existing set of tools is used to illustrate the preferred embodiments of the new tools and applications claimed in the Claims section. The currently preferred embodiment of this invention is based on the definitions of a particular Shell: Gensym Corp 's G2 Expert System. There are several other attributes that relate to the Shell's built-in performance and formatting capabilities, which are not shown in those examples. Some information included within the body of this application was extracted from various sources describing the characteristics of G2, including user manuals (included here by reference), and some of this material is subject to copyright protection.

A. Monitoring and Controlling the Operation of a Reactor Using a Simulated Virtual Model

[0044] The system of this invention is a hybrid combination of model-based methods describing the explicit mechanistic reference behavior of the production system with the input from process sensors that monitor the state of the production system, triggering event-based control flow of operations characteristic of each system. The system implements rules that wait for events to happen. Such events may be complex combinations of individual events, such as selected measured values reaching certain predefined or simulated values, or be within certain predefined ranges, including the implementation of fuzzy-logic within those rules. Such rules may fired other rules or invoke inferencing procedures or cause certain control actions to take place. This system can detect the current status of the production system, and when the selected monitored variables reach certain values specified by the simulated values of those variables, then certain control action take place. The system provides simultaneous supervision of any number of operating variables (such as intermediary or end-products, which in the case of cellular systems can be intracellular or secreted), and compares them with the simulated values of those variables and, depending on the dynamically monitored behavior of the production system, the control system is able to compensate by feeding components at different rates, feeding additional components, or stop feeding some of the previous components.

[0045] Because of the user friendly and intuitive interface. the bench scientists can participate in the design of the production system by editing the knowledge base and the visual models themselves, incorporating information and data obtained in their own experimental research or that from other published resources. It allows the direct incorporation of research into the scale-up operations. The scientists can rapidly build models by selecting any desired building block from the many palettes provided, clone it and drop it on desired compartment, link it to other building blocks in the model, configure it by entering values of desired attributes, and initialize it to establish relationships with other building blocks in the model. The information and data is entered into the system as attributes of objects, either as user inputs, or directly from measuring devices using and interface between said measuring devices and the computer system.

[0046] The system provides a graphical computation and control language, where the objects communicate through the links established between them Some of those linkages are built in within the composite prebuilt building blocks, while the modeler establishes other links between appropriate components from different building blocks. Other information, such as the name, description, references, or the values of parameters specific for each component are entered by the modeler. Variables in this system are themselves objects, and maybe of two major classes: a) one-valued variables have only one value which may be either provided during a simulation by a simulation formula or procedure, or inferred by any other means, such as rules or general formulas and b) two-valued variables have two values: the simulated-value as before, and the measured-value is provided in real time through an external sensor mapped to said variable. The one-valued variables are used by default because of their smaller footprint, and their values are provided by default by generic simulation formulas that are specific for each class of variable. However, the modeler can replace them with the equivalent subclasses of two-valued variables for each instance of a component where the variable is mapped to an external sensor, and when both the measured valued and the simulated value of such variable is desired. The modeler can also write specific formulas for any desired instance of a variable, which then overrides the default generic formulas. The inference mechanisms compare the measured value and the simulated-value, either the current values or the values mapped at some time point in the past, since the system is able to keep a time-stamped history of both types of values, and take specified actions when the inference criteria are met, such as: causing a valve for a component feed to be more or less open or closed, or activating or deactivating whole branches of the model pathways being simulated.

[0047] As shown in FIG.1, a system of this invention is composed of:

a) a reactor (102) which may comprise any combination of reaction tanks, fermentors, bioreactors, or other processing units;
b) one or more data acquisition devices (108) which may comprise any combination of hardware and software devices, such as sets of sensors (106) that measure the amounts of selected chemicals in the reactor, signal transducers, filters, amplifiers, data acquisition boards, appropriate device drivers, or any other required devices;
c) one or more computer systems (112) comprising CPU, memory, storage device, display, user input device, linked (110) to the data acquisition devices (108);
d) one or more computer programs (114) and one or more computer Virtual Models (116) of pathways of chemical interactions and other processes in the reactor (102), wherein the computer programs (114) and the Virtual Models (116) are used to quantitatively or semi-quantitatively simulate in real-time the Virtual Models (116), wherein the

Virtual Models encapsulate one or more variables (118) that represent the quantities of certain monitored entities in the reactor (102) and one or more variables (128) that represent the quantities of certain entities or certain events which when reaching certain values during the simulations of the Virtual Models (116) by the computer programs (114) trigger certain control actions that affect the operation of the reactor (102), and wherein both the computer programs (114) and the Virtual Models (116) are loaded in the memory of one or more of the computer systems (112);

e) one or more monitoring interfaces (120) loaded in the memory of one or more of the computer systems (112), which act as bridges (122) or software interfaces between the data acquisition devices (108) and the computer programs (114) and allow passing of values (124, 126), such as the values of the amounts of certain entities in the reactor (102), as measured (104) by the corresponding sensors (106), to the corresponding variables (118) embedded in the Virtual Models (116) that represent said amounts;

f) one or more controller devices (138) which regulate the operation of the reactor (102), such as controlling the flow of certain inputs (142) to the reactor (102), wherein the controller devices (138) are linked (136) to the computer systems (112);

g) one or more controller interfaces (130) loaded in the memory of one or more of the computer systems (112), which act as bridges or software interfaces between the computer programs (114) and the controller devices (138) and allow passing of control signals generated by the computer programs (114) as a result of the values of any combination of any number of variables (128) embedded in the Virtual Models (116) reaching certain values.

[0048] FIG.1 also shows the directions of flow of data and control in the system of this invention. The amounts of certain entities, which are specific for each particular process design, are captured (104) by the corresponding set of biosensors (106) and, through the data acquisition devices (108), those values are passed (124) to the monitoring interface variables, which in turn pass those values (126) to the corresponding variables (118) in the Virtual Models (116) that represent those values. Those variables (118) are integrated with many other variables and parameters embedded in the Virtual Models (116) during the real-time simulation of the Virtual Models (116), including other variables (128) embedded in the Virtual Models (116) that represent quantities, rates, or other events, and which are monitored by the programs (114) during the simulation. Whenever during the simulation the values of any combination of any number of said monitored simulated variables (128), which are specific for each particular process design, reach certain values, then the programs (114) pass control signals (134), through the appropriate control auxiliary structures (132) in the controller interfaces (130), which are forwarded (140) to the controller devices (138), which in turn control the flow of inputs (142) and regulate (144) the operation of the bioreactor (102), which is being monitored (104) in a continuous manner.

[0049] Depending on the application requirements, the interfaces may provide bridges to Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCS) or Programmable Logic Controllers (PLCs), with the adequate protocol drivers, as well as to relational databases, object-oriented databases, ASCII files, as well as to a number of other connectivity applications, allowing the program to send or receive values over said interface.

[0050] The knowledge-based Virtual Models include model-based reasoning that models the dynamic behavior of processes on the reactor. This mechanistic approach may involve any number of variables to be monitored, including those measured and monitored in the reactor and those simulated and monitored in the Virtual Models. The Virtual Models provide a visual qualitative and quantitative description of processes that happen inside of the reactor, as well as a description of the participants in those processes. The system of this invention separates the representation of the physical systems as Virtual Models from the monitoring and control aspects, allowing to integrate the same Virtual Models with different combinations of monitoring and control designs, to solve different production needs.

[0051] The present invention also refers to a domain-specific, application-independent, framework to construct specific and interactive information, modeling and simulation applications in the chemical and biochemical domains. It provides a variety of tools, graphical interfaces and associated methods to provide users the capability to extract, interactively or automatically, the integrated knowledge contained in those applications or models build by the modeler, and to further use those models, among other uses, to navigate through the pathways of processes and explore those processes and their participants, or for quantitative real-time dynamic simulations. The system comprises diverse sets of objects representing either entities or concepts, hereinafter called bioObjects, and other auxiliary structures, which in general are referred as tools, arranged in object-class hierarchies and workspace hierarchies. Methods may be associated with a class of objects, an individual instance of a class, or a specified group of instances within a class. Libraries of prebuilt knowledge-based generic bioObjects are provided as the building blocks that can be combined in prescribed ways to create diverse and new knowledge structures and models.

[0052] In the description of this invention, the following clarifications should be noted:

a person using the underlaying Shell to define the domain-specific application-independent but knowledge-based classes of objects, their associated methods, and the prebuilt knowledge-based building blocks is hereinafter re-

ferred to as a "developer", while the person using those building blocks to build application-dependent models, or expanding the libraries of prebuilt bioObjects, is hereinafter referred to as "modeler", and the person that extracts and uses the accumulated knowledge and runs simulations, is hereinafter referred to as a "user".
a window refers to a display of the program on a computer screen, while a workspace refers to one of the many containers of objects that may be displayed within a window, and the subworkspace of an object is the workspace that may be associated with that object and may encapsulate the components of that composite object;
the selection of a menu option is equivalent to clicking on that option, and both expressions are used in the following description of this invention;

e) expressions similar to "selection of option A displays B" is a short form meaning something similar to "selection of option A causes the invocation of the action or procedure specified in the definition of option A and, as a result of the execution of such action or procedure, B is displayed in the window from which option A was selected"

[0053] The object-oriented approach provides a powerful knowledge representation of physical entities (such as organs, cells, DNA, enzymes, receptors, ligands, mediators, or ions) and conceptual entities (such as processes and cellular interactions, quantities and rates). Data and behavior are unified in the objects. Objects are defined following class hierarchies in which the definition of each class specifies the types of attributes characteristic of all subclasses and/or instances of that class. Encapsulation permits to hide the details behind each object, and encapsulation is implemented in two different forms: a) at the attribute level, is the standard form of encapsulation of object-oriented approaches; and b) at the workspace level, a less common form of encapsulation related more to the iconic approach. Multiple levels of workspace encapsulation are supported (FIGs 18 and 24), allowing modules with a multilayered structure with increasing levels of detail. The subworkspace is not inherited through the class hierarchy, neither are the components. However, once a generic instance for a class is completed with components in its subworkspace, it can be added to the corresponding palette (FIGs 18, 20, 23) as a prebuilt building block, and the resulting composite object can be cloned, in which case all encapsulated levels of subworkspaces are also cloned. The interpreted domain-specific framework approach is easy to use and allows rapid building of iconic models. The domain-expert knowledge is represented in an easy to understand declarative from, separated from the methods that specified how that knowledge is used by the program. The iconic approach allows the user configuration for building of models to consist primarily of connecting iconic objects and filling out their tables of attributes. The visual framework enables reasoning about the interactions between objects and their compartmentalization. This graphical programming capability allows the continuous expansion and modification of the system, necessary for this type of applications, by scientist modelers with no programming skills. Its modularity allows to delete, modify or create parts of the system without affecting the operation of the rest of the system. The knowledge can also be extended and reused under different contexts, and in different new applications. The same hierarchic architecture used to develop the bioObjects can be extended by the modeler to expand the libraries of bioObjects, creating new objects or by cloning, configuring and modifying existing ones.
[0054] The principle followed in the design and implementation of this invention is the breaking down of the knowledge about entities, processes and pathways down to smaller functional units, to a level where the following requirements can be met: a) allowing their repeated use as building blocks in a variety of situations; b) allowing access to the structures and processes that are susceptible to control and regulation; and c) keeping the number of units manageable. The functional units are composite bioObjects with further components, such as: a) other iconic bioObjects on their subworkspaces, b) buttons to request data displays or input panels, and c) non-iconic objects which are either attributes of other objects, such as domain-specific variables, parameters, lists or arrays. Methods refering to the bioObjects, their attributes, or their components, such as actions, formulas (Tables 1-7), relations, rules (Tables 9-11) and procedures, and which can be invoked interactively by the user from either the bioObjects or their components or from the domain-menus, or invoked programmatically by other rules or procedures, are hereinafter referred to as the associated methods. It is an object of this invention the manner in which the combination of those types of knowledge structures encode, in a distributed form, the data and the knowledge that enables the system to compute and to reason about conditions and events defined by the developer, the modeler, any other external source, and/or generated during a dynamic simulation. As can be observed by comparison with the prior art, the form of knowledge representation and the overall organization of the system object of this invention offer a very different and innovative approach that not only integrates the qualitative and quantitative description of chemical and biochemical objects with a set of state and dependent variables, such as amounts and rates, incorporates temporal reasoning and generates dynamic simulations. Composite bioObjects represent either:

a) the descriptive characterization of single units of molecular entities, molecular subcomponents, or molecular complexes, called bioEntities (FIG.18);

b) the description of a population of such molecular entities or molecular complexes in specific states, called a bioPool, which together with the potential inputs to and outputs from such bioPool is encapsulated in what is hereinafter called a bioReservoir (FIG.8);
c) the interactions between fractions of different such populations of molecular entities or molecular complexes. or their transitions from one molecular state or discrete compartment to another over time, called bioProcesses (FIG.7).
d) the organization and integration of networks of pathways composed of sets of linked bioReservoirs and bio-Processes (FIGs .7,8) organized in compartments hereinafter called bioModels (FIG.24). Low level bioModels can be further organized in discrete location and time compartments, representing the subcellular organelles and the phases of the cell cycle respectively, which are themselves components of higher level compartments representing single cells. All those types compartments are themselves bioModel subclasses at different levels of complexity and detail in a hierarchy of subworkspaces.

[0055] BioObject, a subclass of the class objects provided by the Shell, is the superclass at the top of various hierarchies of classes of objects such as the class hierarchies of bioEntity, bioReservoir, bioProcess, bioModel, and others. Each instance of a class is characterized by the set of attributes defined for the class, which can be inspected in the table of attributes attached to each object by selecting it with the mouse. Some attributes define configuration information, while other attributes describe the composition and characteristics of the object, and still others hold dynamic state information, such as the current value(s), data histories, and status. The value of some attributes of an object, such as the variables and parameters, can be dynamically modified at runtime.
[0056] Model-based knowledge is integrated and encapsulated in the structural, functional and behavioral models represented by the virtual bioModels, defined qualitatively (FIG.24) by the locations, connections, and relations of their components, and quantitatively by their encapsulated mathematical models (FIGs 12 - 15). The constants, parameters and variables that define those models are distributed through the different types of bioObjects, defined as their attributes, and the corresponding formulas and functions relate those data structures to others and characterize the particular system.
[0057] In the current implementation of this invention, methods such as procedures, rules, formulas and relations, defined in a structured natural language, may apply to a single instance or a group of instances, or to a class or a group of classes, and they may include references to connections and to connected objects. Methods may refer or apply to the values of attributes at different time points or to the behavior of variables or parameters over time. They can perform in response to: a) given events or conditions, b) at predetermined time intervals, or c) upon request from other rules or procedures. Methods can be executed in real-time, in simulated time, or as fast as possible, implementing different strategies concurrently and over extended periods of time. Arithmetic and symbolic expressions can be used independently or combined, and dynamic models may include from non-linear differential equations to logic expressions to simulate both analytic and/or heuristic behavior.
[0058] The following sections will also discuss various of the innovative teachings of this invention which refer to methods and tools used to retrieve and display the stored information and data, encapsulated in graphic and interactive models of complex molecular mechanisms and pathways, allowing to:

a) interactively navigate through those pathways (FIGs 31, 32, 35, 36);
b) perform queries that refer to: a) the functional modular structure of the bioEntities involved, b) the position of bioPools downstream or upstream of the bioPool taken as reference, c) to the location in cellular compartments of the processes in which those bioPools of bioEntities participate, d) any combination of the previous three (FIG. 30); and
c) dynamically simulate the kinetic interactions between the bioReservoirs and bioProcesses, to mimic the regulation and function of cellular systems (FIGs 33, 34).

[0059] The system of this invention can be implemented layered on top of any advanced real-time object-oriented expert-system development environment or shell that support the methodologies described throughout this application, or it can also be developed in total or in part from scratch, parting from any combination of hardware and software platforms, either already existing or developed for this or other purposes, that provides the capabilities and means required for this implementation. While many of the underlaying capabilities referred to below are common to other static or real-time expert-system development environments, some capabilities may be currently specific for G2, but equivalent capabilities can be expected to be developed by others. Among the capabilities provided by the Shell, are: structured natural language, a graphical language, supported by an icon editor,an inference engine, a simulator, which may compute the values for dependent variables from algebraic equationsdiscrete-state variables from difference equations: continuous-state variables from differential equations, a supervisor which controls operations such as task prioritization, asynchronous concurrent operations. and real-time task scheduling, an inspect facility.a describe facility, a tracing and debugging facility, and support for distributed applications.

[0060] The Shell's class object-definition provides templates to define the attributes of new classes of objects, such as: the names, types and default values for new attributes for that class, an icon specific for that class, and stubs that define the types of connections. Instances of the so defined classes can be createdprogramatically, in which case they are transient but can be made permanent; or by selecting the "create-instance" option from the menu of the object-definition. Additional instances can be also created by selecting the "clone" option from the menu of a previously created instance.

[0061] Each variable has an attribute called data-server, which value determines how said variable receives its value: from the inference engine, the simulator, or interfaces to external monitoring devices, databases, or file systems. By default, in he current implementation of this invention, the variables are simulation variables, which means that their data-server is the simulator. A simulation variable may have two values: the current value may be optionally provided by inference engine or one of the interfaces, while the simulated value is provided by the simulator. In those instances where the variables correspond to substances that are being monitored in the reactor, the variables are then set to receive two values, the current measured value, provided by the external sensors through the appropriate interface, and the simulated value. The non-simulated values may be set to expire after a desired period of time, so that the inference engine seeks a new value when needed. Different part of the simulation may run at different times, as desired.

[0062] The current implementation of this invention also allows the inclusion of fuzzy-logic statements, such as if (y < 6 (+-3)) /= (z > (+-2)) in the inference engine, while the simulator only uses discrete values. Fuzzy operators comprise: is more true than, is less true than, is not more true than, is not less true than, =, and /=. The fuzzy truth values are assigned to expressions with a fuzzy truth band in parenthesis, such as: the concentration of receptorI > 0.8 (+-0.2). The truth-threshold is set for the inference-engine as a whole. For example, if said threshold is set to 0.7, for the antecedent of a statement to be true it must be greater than or equal to 0.7. This invention improves the control of the sequential addition of inputs to the reactor, at times and in amounts that are appropriate for the current status of the biochemical processes within the reactor. In the examplebelow, it is expected that the concentration of receptorI expressed on the cell surface, which is available for detection by the receptorI-sensor, is induced at some point by some internal events that happen inside of the cell. Such internal events are the result of complex interactions, induced by the occurrence of some combination of previous events, including those events induced as a result of some previous addition of some other ligand, or any other chemical or combination of chemicals, controlled by one or more other pumps controlling inputs to the reactor. This fine control based on the knowledge about the mechanisms involved avoids, for example, to add costly ligandI to the reactor at a time when the cells are not ready to use it, with the likelihood that it will be destroyed by other enzymes, and the risk that it will not be active when the cells become ready to use it at a later time. It also could prevent that ligandI exerts damaging effects to the cells when provided when cells are in the wrong state, as it has been demonstrated for various compounds that induce programed cell death when signal from said compounds are transmitted to the cell in the absence of other required signals. An example of such control statement using fuzzy-logic is: "if the average over the last 3 minutes of the concentration of receptorI > 0.8 (+-0.2) and the average over the last 3 minutes of the concentration of ligandI < 0.5 (+-0.1) and the average over the last 3 minutes of the flow-of-pump-L1 = the flow-of-pump-L1 then conclude that the flow-of-pump-L1 = the flow-of-pump-L1 * 1.2 and call adjust-flow-of-pump(flow-of-pump-L1)", wherein: a) the concentration of receptorIand the concentration of ligandI are variables coupled to the corresponding sensors for those substances in the reactor; b) the statement "the average over the last 3 minutes of the flow-of-pump-L1 = the flow-of-pump-L1" prevents additional changes in the flow for an specified period of time, to allow the system to adjust to the last changes; and c) adjust-flow-of-pump(flow-of-pump-L1) is a remote procedure call that activates the external device that controls the pump controlling the delivery of ligandI, while passing the value of the variable flow-of-pump-L1 to it.

[0063] In an alternative implementation, the bioPools of the components that regulate critical cellular events, or of those that are monitored in the reactor, may have an additional attribute defined as a fuzzy logic parameter which value can then be accessed by different monitoring/control statements. Fuzzy values may take values from -1.000 true (certainty of being false) to 1.000 true (certainty of being true). The value of said parameter could be provided by a rule, or by a generic or specific formula defined by the modeler, which is dependent on either the measured or simulated value of the corresponding concentration variable. For example, said fuzzy attribute may be called Availability, and its value could be provided by a rule, such as: "if the concentration of any substrate S > the Michaelis-constant of S (+- the Michaelis-constant of S/2) then conclude that the availability of S is true". The monitoring/control rules could then refer to such fuzzy logic variables being true or false, and would allow the modeler to modify the rules for single or groups of bioReservoirs, without the need to modify the control rules, such as: "if the average over the last 3 minutes of the availability of receptorI is true and the average over the last 3 minutes of the availability of ligandI is not true and the average over the last 3 minutes of the flow-of-pump-L1 = the flow-of-pump-L1 then ... "

[0064] The current implementation of this invention also allows to:

a) use the simulated Virtual Model to entirely take control of when and how the inputs are added to the reactor, (or the outputs, such the culture medium or cells, removed): or

b) synchronize the simulated model to the processes in the reactor, by allowing certain critical measured values from the reactor to provide the values for their corresponding variables in the simulated model, which are then used by other dependent variables and propagate down the simulated model.

B. The Virtual Model: Domain Specific Knowledge Representation

[0065] The computer system, methods, and interfaces that are the objects of this invention can be applied to visually model and quantitatively simulate any type of complex system involving any type Qf processes and their participants. The set of palettes of prebuilt building blocks provided should reflect the needs of the particular domain in order to facilitate the tasks of the modeler, permitting the basic paradigm of "Clone, Link, Configure and Initialize" and other automated methods. A library of prebuilt Query Panels and other panels, such as Navigation, Simulation and Experiment Panels facilitate the tasks for users to rapidly extract and dynamically use the knowledge and data contained in each specific application. The description that follows is based on one particular domain, that of biochemistry, to illustrate one of the implementations of such computer system, methods, and interfaces. However, similar computer systems, methods, and interfaces can be implemented in a similar manner for domains other that of biochemistry. For example, in a business domain, what in this description is called bioEntities can be substituted with different types of entities, such as employees or projects, and bioProcesses can be substituted with the different activities in which both interact, the bioReactants with the roles they play in such activities, which frequently may involve other participants. Employees are like enzymes, with the more capable having higher rate-constants. Projects are like the substrates, and employees may have higher affinities for some projects, with a likelihood that they will interact in a process in which a quantifiable measure of the outcome, such as quantity or level of quality, will depend on quantifiable measure of the employee, lets say amount of time put on that project and level of specific experience. The bioPool of each employee could represent his/her time, and the max-amount could be set to a day, a month, a year, or any other amount. The Contribution to each project would be the amount of time invested in that role, while the different activities will compete for the time left available.

[0066] In the remainder of the description of this invention, the focus will be on biochemical systems, which are complex, hierarchical, heterogeneous and non-linear systems, involving an interplay between the processes of transport, reaction and conformational change. They are regulated by cybernetic flows of information. This invention uses a kinetic approach to model these chemical and biochemical systems, rather than a thermodynamic interpretation, although thermodynamic variables are also included to allow modification of the behavior of the kinetic system. The current approach, which focus on the amounts of entities, such as concentrations, densities, scaled-amounts, or other quantities of each bioPool of cells, molecules, or molecular complexes, in a particular state, as a continuous function of time, and on specific coefficients or rate constants that, in conjunction with the current values of those amounts, define the velocity of a bioProcess, and the rates of consumption of certain bioReactants that participate in such bioProcess or the rates of production of its bioProducts, which are dependent on such velocity and specific stoichiometric coefficients, and which are equivalent to the rates at output and input from their connected bioPool, respectively. This kinetic approach is closer to the way of thinking of biochemists.

[0067] The cellular representation in this invention aims to reproduce the fact that function emerges from the cooperation of many components, which depends on compartmental organization and functional relationships, and deals with multiple levels of biological and biochemical structural complexity, such as physiological systems, organs and their compartments, cells, subcellular organelles, and molecules. The schematic representation on the left side of FIG.2 shows an abstraction of the types of molecular and cellular relationships and interactions that are modeled and simulated, with unlimited levels of encapsulation allowing to start with a high-level representation, and then successively "clicking" into objects of interest to show the next level of detail. This high-level example can be summarized by what a biomedical expert could describe as "Oncostatin M (201) induces different responses on different cell types: a) it induces certain classes of cells (202) to secrete the cytokine IL-6 (203); b) it induces differentiation of primary, coblestone-like AIDS-KS cells (204) into mature spindle-like AIDS-Ks cells (205), which are capable of long-term growth in agar when c) both Oncostatin M and IL-6 are present". This statement and its graphic representation combine knowledge at very different levels of detail, equivalent to the way human mind can abstract knowledge. For instance, it deals with both molecules and cells as entities as if they were similar types of structures, when that is very far away from the reality. Cells are composed of thousands of different types of molecules. In some cases, an overall pathway within the cell can be known to some extent while in others the mechanism of a biological response may be totally unknown.

[0068] The system of this invention can be described in very similar terms. On the one side, bioPools of cells and molecules can interact with each other, in any combination in a variety of bioProcesses, as shown in FIG.29. On the other side, as schematically shown on the right side of FIG.2, the components of each type of cell can be modeled and analyzed at increasing levels of detail, as encapsulateed in various layers of bioModels (202 - 212, 216, 217), until at the end of the layer hierarchy (213) the single bioProcesses (214) are represented. and their associated bioReservoirs

(215), which point to their associated bioEntities (not shown). In general, the signals transmitted by the cellular signaling or regulatory pathways progress within each cell in repeated cycles, driven by external and internal events, and sometimes time, in a synchronous way, and they may extend from cell to cell, sometimes activating the producer cell as well. A typical but simplified high-level example of a cellular signaling pathway driven by a certain agonist, such as Oncostatin M, is shown in FIG.2 as: agonist (201) → receptor activation (206) → signal transduction (207) → gene transcription (208) → mRNA processing (209) → protein synthesis (210) → secretory protein processing (211) → transport → (receptor activation). By clicking on objects at each level, one can focus on specific areas at the desired level of complexity. To represent biochemical systems it is necessary to refer to the characteristics and current state of single entities, and at the same time to refer to populations of entities with similar properties. In a small application for molecular genetics, Karp (1989) was only able to represent and implement the overall system either as a network of molecules interacting in a pathway or as a network of variables quantitatively representing the processes in a pathway, as separated systems, but he was not able to integrate both. This integration is one the teachings of this invention. Furthermore, contrary to the system of this invention, his representation was static and did not include temporal reasoning, having all processes represented as instantaneous reactions.

[0069] FIG.3 shows how the intermediate submodels are preferentially organized in two types of superior compartments (301) of a cell represented by the different subsequent cell states represented by the cell-phases (302 - 308), and within each of those the cell-compartments (309 - 316) that represent the different cell organelles. Each of the cell-phases contain a detailed description of large numbers of components, encapsulated in the different sublayers, and a set of characteristic parameters and time-variant attributes (317 - 320) that can be used to model the progression of a population of cells - those compartments. That concept is schematically represented in FIG.4, where some of the cells of that population X1 may keep cycling through those phases (401), as represented by the bioPools of cells in each phase: Go (404), $G_{1.1}$(406), $G_{1.2}$ (409), S (411), G_2 (413), and M (416), if some specific and required conditions are met, as specified by their corresponding transitions represented here by the translocation-bioProcesses (405, 408, 410, 412, 415, and 417), or they can exit the cycle if other set of conditions are met (407) and differentiate to another related cell type characterized by different functions and therefore a different cell cycle (402, 403) or if yet other set of conditions are met (407) they may dye by programmed cell death or apoptosis (414).

[0070] A cell-bioModel is represented as a set of bioReservoirs and bioProcesses, systematically organized in the subworkspaces of other bioModels contained in such cell-bioModel, which represent each a separate compartment in time and location, and within those, sets of biologically related processes or pathways. At a higher levels of organization, bioProcesses connected to bioReservoirs of cells or cell-interactions, are organized into bioModels that represent organ compartments, which are organized into organs, and which are organized into into physiological systems. Different cell-bioReservoirs encapsulate either populations of different cell subsets or populations of the same cell subset in different stages or in different locations. Each of those cell-bioReservoirs makes reference to a predefined cell-bioModel, which has a characteristic phenotype representative of the population of cells encapsulateed in that cell-bioReservoir. The different phenotypes may characterize: a) different cell lineages, b) different stages of differentiation within the same cell lineage, or c) cyclical changes in the cell characteristics over time, within the same differentiation stage. Phenotypes are sets of characteristics measurable on intact cells, and the transition of cells from one phenotype to another are represented in the present embodiment of this invention as a translocation of a fraction of the cells from the cell-pool of the former subset to the cell-Pool of the later subset. The events that trigger those transitions may be unknown, but they may be recognized by changes in measurable or functional markers, such as the appearance of a new receptor, the synthesis of DNA, the secretion of specific interleukins, or mitosis, and the relative time in which they appear or disappear may be known, in which case those transitions are time-driven. More specifically, a cell-bioEntity, which is a high-level external representation or description of a cell, may make reference to a cell-bioModel, which is internally characterized by its components and mechanisms, represented by the pathways of bioReservoirs and bioProcesses.

[0071] FIG.5 is a diagrammatic introduction to important concepts upon which much of the processing and reasoning of the current invention is based, where the different layers of arrows represent the direction of flow of information and data. In this system, pathways refer to sets of alternating layers of bioReservoirs and bioProcesses connected to each other to indicate participation of the populations or pool of chemicals represented by the bioReservoirs in the processes represented by the bioProcesses, and to further indicate that the processes represented by the bioProcesses provide inputs to the pool of chemicals represented by the bioReservoirs. The result of those specific connections is a multidimensional network of interconnected biochemical pathways that determines the basic architecture of this system. That basic architecture also defines the resulting network of parameters and variables, and the specific arguments for the generic formulas that provide the values for those variables are determined by those connections. A **bioReservoir** (501) is an iconic knowledge-structure that encapsulates a **bioPool**, which refers to a population of a single type of molecule or complex contained within an imaginary space. A chemical process is represented as a **bioProcess** (502), an iconic knowledge-structure that encapsulates various components, including a **bioEngine** (504) that controls the Velocity at which the input(s) of the bioProcess, the **bioReactant(s)** (503), are consumed and the output(s), the **bio-**

Product(s) (505), are produced. BioReactants can also be said to represent the participation of certain number of units of a bioPool as the input to a given bioProcess, and more specifically the role that such number of units of the bioPool play in such bioProcess. BioProducts can also be said to represent the participation of certain units of a bioPool as output from a given bioProcess Both types of knowledge-structures integrate bioProcesses with bioReservoirs in a complex architecture that allows to model the kind of complex systems and multidimensional pathways that are characteristic of the domain of this invention. In such integrated system, units from the bioPool of a consumable bioReactant, are transferred to the bioPool(s) represented by one of its bioProduct(s), at a dynamically computed rate. The term upstream is used hereafter to indicate those bioReservoirs, bioProcesses, or pathways that affect the inputs to a given bioReservoir or bioProcess, while the term downstream is used hereafter to indicate those bioReservoirs, bioProcesses, or pathways that are dependent upon the outputs to a given bioReservoir or bioProcess.

[0072] The fast and short-term regulation, such as the temporary inhibition or activation of enzymatic activity is modeled through separate bioPools of the more active and less active (or inactive) forms. Each modification of molecules or complexes that results in qualitative or relevant quantitative changes in their activity or function, is represented as transfer of units between those different bioPools. Examples of those modifications include post-translational modifications of proteins, including allosteric changes, such as phosphorylation, isoprenylation, and so on. The slower, long-term, regulation of enzymatic activity is modeled by induction or depression of protein synthesis. which optimizes its concentration for the newly required function. Constitutive enzymes and receptors are considered to be synthesized and degraded at a constant rate, resulting in a constant steady-state level. In regulated molecules, environmental signals such as the extracellular availability of a hormone or growth factor may cause the rate of synthesis or expression of new surface receptors to increase X-fold. If the rates of outputs are not concurrently and equally regulated by the same factor, then a new steady-state level will be reached, which may or may not return back to normal after the activating signal ceases. Examples of those modifications include: a) gene mutations and other modifications, b) DNA transcription, c) post-transcriptional splicing and other modifications of RNA, and d) translational regulation

[0073] BioPools, bioReactants, bioEngines, and bioProducts are classes of basic bioObjects that further encapsulate a variety of knowledge structures, including quantitative attributes, parameters and variables, which values, provided by simulation formulas, can be dynamically simulated in real-time. The amount of such units comprised in a bioPool, represented by a variable such as Concentration or other appropriate amount, is dynamically computed at run time, parting from an initial value or basal-amount set by the modeler, or in its absence, from a default value set by the developer. The arguments of the variables of these basic knowledge structures are the output values of certain other variables and/or parameters that are attributes of either: a) the same bioObject instance or b) instances of connected bioObjects of different classes. For instance, an argument for the Consumption-Rate of some bioReactants (503) and for the Production-Rate attribute of any bioProduct (505) is the value of the Velocity attribute of the connected bioEngine (504), while the values of the Contribution attribute of each bioReactant may be arguments of the Velocity attribute of the bioEngine. Furthermore, the Production-Rate (506) of some bioProducts and the Consumption-Rate (507) of some bioReactants are arguments for the Input-Rate and the Output-Rate attributes, respectively, of the bioPool to which they are distantly connected, while the argument of the Contribution of any bioReactant (503) may be the value of the Concentration attribute of the connected bioPool. Told in a different way, note that: a) the flow of data between the bioPool and a bioReactant is bidirectional, involving two variables in the bioPool and two in the bioReactant, b) the flow of data between a bioReactant and the bioEngine is bidirectional, involving two variables in the bioReactant and one in the bioEngine. c) the flow of data between the bioEngine and a bioProduct is unidirectional, involving one variable in the bioEngine and one in the bioProduct, and d) the flow of data between a bioProduct and the bioPool is unidirectional, involving one variable in the bioProduct and one in the bioPool. The reversibility of any binding can be represented in two alternative ways: a) implicitly reflected in the output of a single process, which is the balance in the predominant direction (as represented by the specific or global kinetic parameters). or b) explicitly modeled as two separated processes in the forward and reverse directions.

[0074] The value for the basal-concentration, basal-density or basal-scaled-amount parameters of each population reflects the physiological steady-state value, when the rate of change is equal to 0 or equivalently when the sum of the inputs equals the sum of the outputs. Concentrations are used in general within classes of bioPools representing soluble entities. However, as it is found empirically in biological systems, most most entities are associated with membranes, with large polymerized structures such as the cytoskeleton or the extracellular-matrix, or associated with one or more other macromolecules forming complexes. This means that the concentration of those bound entities is less relevant than the actual amount of those molecules that are available in the appropriate compartment at a given time, and therefore the term density is used to represent units per compartment. Furthermore, most simulations of biological systems are more meaningful in terms of relative amounts or ratios among the quantities of interacting or regulatory molecules, and therefore the alternative variable called scaled-amount is provided for each bioPool, which is dimensionless and does not require units. The values of the Velocity of the generic bioEngines, as provided in the Palettes, are computed using the default generic simulation formulas associated with the subclass of parameter or variable associated with it, which by default are dependent on the values of a set of scaled variables and parameters. The user can however replace them with instances of other velocity-pvar subclasses from the library provided each with its associated generic simulation formula based on absolute values (Table 3), depending on the subclass.

[0075] Life and growth depends upon closed cycles of mutually dependent interactions, and the steady-state correspond to an optimum state. since the lack of such balanced state would lead to rate-limiting steps. The principle underlaying this system's dynamic modeling is the network of a combination of state and dependent variables, encapsulated within the structure of the bioObjects contained in the specified bioModel. The bioObjects provided are programmed by default for steady-state modeling. A dynamic simulation is initiated after the introduction of desired perturbations or initial conditions by the user, and the input data initiates a forward chaining which involves both control and data flow from bioReservoirs to its connected downstream bioProcesses, and from these to their connected downstream bioReservoirs, moving along the bioModel's pathways. Inputs from external control systems or databases can be also forward-chained during run-time. Other parameters and variables defined in the system of this invention are also involved in the dynamic quantitative simulationsand can be incorporated in two distinct and alternative forms of simulation reasoning, absolute or scaled. For example, the factors that affect and define the transformation rates, such as the catalytic, binding or transfer rate of a bioProcesses (here preferably encapsulated as the Velocity attribute of its bioEngine), are distributed as attributes of the various connected bioObjects and comprise:

a) the quantities of the bioReactants of different types, such as enzymes, substrates, inhibitors, and activators, or receptors, agonists and antagonists, or components of a complex (here preferably encapsulated as the Concentration, Density, or Scaled-Amount attribute of the bioPool connected to each bioReactant, which values can be directly incorporated into the equation that gives the value of such Velocity, when using the absolute reasoning; or indirectly incorporated into the equation that gives the value of the Contribution attribute of each bioReactant, which holds an intermediary scaled value that is incorporated into the equation that gives the value of such Velocity, when using the scaled reasoning);
b) the values of specific kinetic parameters characteristic of each type of bioReactant (here preferably encapsulated as: the Effective-binding-sites attribut e of enzymes and receptors, and the Michaelis-constant, Equilibrium-dissociation constant, or equivalent attribute of substrates, ligands, or other bioReactants, respectively, which values can be directly incorporated into the equation that gives the value of such Velocity, when using the absolute reasoning, or, alternatively, the Scaled-Michaelis.k, Scaled-equil.dissoc.k, or equivalent attribute of substrates, ligands, or other bioReactants, respectively, when using the scaled reasoning), or, alternatively, a global kinetic parameter specific for that bioProcess that replaces and represents the combination of the specific parameters of each of the bioReactants of that bioProcess (here preferably encapsulated as the Rate-constant-sec attribute of its bioEngine):
c) the stoichiometric coefficients (here preferably encapsulated as the Stoichiometric-coeff attribute of bioProducts and bioReactants); and
d) the effects of environmental conditions such as temperature and pH are also specifically defined as attributes, and can be integrated into the simulation within the appropriate equations.

[0076] Chemical and biochemical reactions are continuous processes, within the time intervals that are dependent on the type of reaction and the environmental conditions, with the balance of the reaction favoring in general the forward direction. In the preferred embodiment of this invention, the specific way the bioObjects are connected specify the sequential or concurrent ordering of data flow, and the parameters of model equations are included as object attributes. Model-generated results may be used as input data for other simulated equations or as events to be used by the inference engine. These dynamic models can also reason about the behavior of variables in the past and project the behavior of variables into the future. Integrating the activities of the different parts of a model system requires the simultaneous solution of a set of non-linear ordinary differential equations and a set of dependent algebraic equations. In this invention, this system takes advantage of the two methods of explicit integration supported by the Shell: either the first-order Euler scheme or the quadratic fourth-order Runge-Kutta scheme, depending of the desired accuracy of the computation. The updates of the different variables may be synchronous or asynchronous, as a result of the variables having equal or different update intervals. The dynamic values of variables or parameters may be displayed in both digital and graphic forms.

[0077] The system of this invention deals with the "incomplete information" characteristic of biological systems by integrating the scaled or semiquantitative values with the absolute values. The scaled-valued variables, such as scaled-amount, have values within the 0.0 to 100.0 scale to normalize the diverse ranges of magnitudes involved in the system. The default initial values of the scaled-basal-amounts vary within this range to best represent the knowledge about the physiological steady-state conditions. As a way of examples: a) a value of 100.0 may model an constitutive abundant enzyme, receptor, or complex-subunit in their inactive form, while this value may decrease at run-time as the value(s) of their active or other derivative(s) increase, so that the total remains 100.0; b) a value of 50.0 may model the normal steady-state catalytic concentration for a constitutive regulated active state of an enzyme, or the steady-state concentration of a substrate, binding protein or ligand, or c) a value close to 0.0 may model any highly regulated induced molecule.

[0078] **Temporal reasoning** is important to model and describe the causal mechanisms that drive biological systems. Temporal reasoning is achieved in the system of this invention in several ways:

a) In general, temporal reasoning is implicit in this real-time system during a simulation, when the values of the variables and many of the parameters vary over time.
b) Entire contents of bioModels, such as those representing the successive phases of the cell cycle, and other time-submodels within those phases, can be activated and deactivated over time by activating or deactivating the subworkspace of such bioModels, as specified within procedures and rules, at pre-specified simulation time intervals, or based on events generated at run-time, such as a certain variable reaching a threshold value, or a combination of both
c) The subworkspaces of specific bioProcesses can be also programmed to be activated for a specified period of time after their activation, as given by its activation-hold-interval attribute. This attribute can be a default value or a value entered by the user or as modified at runtime. This is an alternative to having the control of the inactivation of such subworkspace based on another event. It can also be reasoning by a combination of a given time interval or a given event, whatever comes first, or by events triggering the change of the value of the activation-hold-interval attribute in a pre-specified manner. One such option is to provide a tabular function such as by looking.

[0079] In the currently preferred embodiment of this invention, a unit of a given chemical entity, referred to as a bioEntity which describes the characteristics and state of a single molecule or complex, is considered a separated knowledge structure which is referred to as an attribute of a bioReservoir and can be accessed from the menu choices of this and other bioObjects, such as a bioReactant connected to its bioPool, which refer to a population of molecules with the characteristics described for such bioEntity. The information about the characteristics of particular functional states of a given molecule or complex is encoded as separate instances of classes of bioEntity. Different behaviors of each bioEntity are then explicitly defined through the mechanisms represented by the bioProcesses in which bioReactants that make reference to such bioEntity participate. In other words, it can also be said that bioProcesses represent the interactions of a bioPool of such bioEntity with bioPools of other bioEntities, as defined visually through their connected bioReactants or bioProducts. Furthermore, in the present invention, a single molecule can be described in more detail by and ordered and connected set of instances of more elemental bioEntities, referred to as bioEntity-components, which represent molecular functional components such as subunits, domains, motifs, active sites, regulatory sites and mutation sites. The additional levels of detail are implemented in this invention through the **encapsulation** of subworkspaces. In a similar way, sets of simple molecules can be components of other bioEntities representing a complex, or of a heter-mol-complex. Since unlimited levels of encapsulation are allowed, it is possible to start dealing with a topic with a very general visual overview, and then successively "clicking" into objects of interest, to show the next level of detail.

[0080] In the system of this invention, the methods used to process the application-specific knowledge are separated from the application-specific knowledge itself. The domain-specific but application-independent knowledge is encoded in at least four different forms, characterized by: a) the selection, combination and specific connections of the simple knowledge structures that contain the variables and parameters that quantitatively define this system, within the subworkspace of a prebuilt composite bioObject; b) the design of those simple structures and their interactions as components of the composite structures, which provide for the resulting transparently interconnected architecture underlaying this system; c) the selection and definition of the specific attributes of each of those types of structures, which describe and characterize the potential behavior of those structures; and d) the methods in the form of procedures, rules, formulas, functions and relations that together provide for the monitoring, control and execution of the system. On the other hand, the application-specific knowledge is incorporated in at least two different ways: a) the selection, combination in different ways and connections of those prebuilt composite bioObjects and predefined structures; and b) the configuration of the values of the predefined attributes of those selected structures.

C. Operating Facilities for Developing, Modeling, Navigating, Queering, and Simulating the Virtual Models

1. Developer Mode: Definition of the Building Blocks

Domain-menus

[0081] The domain-menus associated with this mode in this invention facilitate the discussion of the organization of the definitions of object classes and methods. Selecting the "Help & Mode Menus" head pulls down a set options, and selecting any of the available user modes changes the mode of operation and displays the specific domain-menus associated with that user mode for the window from which it is selected. Selecting Developer Mode displays the additional menu heads: "Object Definitions" with pull down options pointing to major groups of object classes, "Variable Structures", "Rules Proc's & Relat's", and "Formulas & Functions". For example, selecting "bioEntities" displays the pull-down menu which options refer to different subclasses of the class bioEntity, and further selecting "MoleculComponents" and "Protein Motifs" displays a workspace that contains the definitions of the class protein-motif and all its subclasses. In a similar way, selecting "Variable Structures" and further "Variables" and "Velocity pVariables" displays the workspace containing the definitions of the class velocity-pVar and its various layers of subclasses.

General Groups and High-Level Classes

[0082] The classification and definitions of the object classes is now described. The different classes of objects and groups of tools and methods comprised in this invention are all knowledge structures implemented as objects. Those tools and methods are preferably organized into major groups, each comprising several object classes, which may each further comprise several levels of more specific subclasses, to be discussed in more detail in the following sections. The class bioTool a subclass of the class object defined within the Shell, is defined to include a variety of auxiliary iconic structures used in this invention, and has no additional attributes. Each top class of those groups is a subclass of one of the generic classes of objects defined within the Shell, from which they inherit a set of attributes and capabilities. Several of those capabilities may be proprietary technology specific for that Shell. For further details, the reader is referred to the G2 Reference Manual Version 3.0 or Version 4.0 and other documentation produced by Gensym Corp., which are here included by reference. In this filing, only those Shell-defined attributes that are necessary to describe this invention will be considered, and these attributes or their equivalents may be common to other development environments. Other attributes and capabilities not mentioned here may, however, facilitate the development tasks and improve the performance of this invention. In the following discussion, attention will be focused mainly on those attributes incrementally defined for each subclass.

[0083] There are several other basic classes defined within the Shell that the system of this invention uses without relevant modifications. One of those are **workspaces**, which are items used to hold and display other items. and which attributes include names, user restrictions, margin, border, background, and module assignment. Workspaces can be independent higher-level workspaces, or subworkspaces subordinated to objects. Subworkspaces may also have the capability of being activatable, if so defined in their superior object, which means that the objects upon them are only active and accessible to the inference engine and the simulator when such subworkspace is activated. Other capabilities of the Shell, such as flashing of the icons, changing colors, or rotation of specific patterns can be used to animate a simulation and to indicate relevant events. such as showing the bioObjects that are activated or which variables' values are out of range, and other applications.

[0084] The object class definitions include attributes such as: Class name, Superior class, and Attributes specific to class. Other attributes are mentioned only when relevant or modified. In addition to an attribute table and sometimes a subworkspace, each iconic object has an associated menu with a set of options. Some of those options perform standard tasks common to all objects, such as: change-size, clone, color, delete, describe, disable, go-to-subworkspace, move, and name, which are self-explanatory. Other novel and domain-specific menu options are defined in this invention to provide interactive access by the user to automatic or semi-automatic tasks related to the configuration of the knowledge structures and to requests diverse forms of processing and display of such knowledge. User-menu-choices are defined for the specific class named in its applicable class attribute, and appear as options in the menu associated with the icon of every instance of every subclass, unless a class restriction is defined for any of those classes to exclude that option from the menu in any particular user mode. Selecting any particular options triggers one or more actions. Those options and the methods invoked upon their selection will be discussed in later sections. Here we will focus on the definitions of all object classes and on the description of their attributes.

Variable Structures:

• Variables and Parameters:

[0085] In the present invention, the quantitative attributes of object classes may be defined as simple attributes, or as instances of subclasses of parameters or variables. The values of simple attributes can be modified through rules, procedures, or other actions of the inference engine. The values for the variable and parameters may be provided from different sources. The Shell's simulator is used in the current implementation to compute the simulated values of variables and parameters that are integral parts of the bioReservoirs and bioProcesses, including state variables. Those values are preferentially provided by generic simulation formulas, although modelers may also provide specific simulation formulas for specific variables. Users can also input values for the simulator through the input-panels described below, and the program can also update values based on values received from other sources, so that values from different sources are integrated. To standardize the units used during a simulation, the absolute values are preferentially entered in Molar, M/sec, moles/liter, or seconds as pertinent.

[0086] There are large number of subclasses of parameters and variables involved in this system. Only a few of them are used as independent parameters or variables represented by their icon, and usually they have auxiliary functions. Most parameters and variables are implemented as attributes of the various classes of bioObjects, or other objects such as model-blocks (FIG.19) or inference blocks (Table 8), as well as attributes of other variables. Aside from the high-level classes, most of the other subclasses do not have additional specific attributes, but many have distinctive default settings, as specified in their listed definitions. The subclasses allow to attach specific methods to each class, either in the form of formulas, or by referring to all members of the class in rules, procedures or other actions, and only a few that are representative or important to the understanding of the operation of the overall system will be discussed in the relevant section. Most domain-specific variables and parameters are subclasses of various subclasses of the the Shell's class parameter, such as float-parameter, integer-parameter or symbolic-parameter; text-parameter or truth-parameter. Those that represent parameters in this domain are grouped under the domain sub-classes int-par or float-par, depending on their value types. Those that represent variables in this domain are grouped under the domain subclass float-pvar, which are subclasses of float-parameter. This invention also defines a set of subclasses of float-var, a subclass of the Shell's class float-variable, as an alternative for domain attributes such as Concentration, Density and Velocity (which by default are represented by pvars or parameters such as a variety of kinetic constants (which by default are represented by simple float attributes).

[0087] The Shell's "variables" have capabilities important for the implementation of this invention such as: a) allowing the user to set a desired instance, lets say the concentration or density attribute of a bioPool representing an entity which quantity is monitored in a reactor, to receive a measured value from the external monitoring sensor while its simulated value is being simulated in parallel, based on the Virtual Model specifications, and then by using inference rules to monitor either value or both values, and to compare them to a specified threshold or range or to each other, and to initiate some control actions to regulate the reactor system monitored by the sensor; and b) allowing the user to define simulation or general formulas specific for a particular instance of a variable by writing the formula directly in the corresponding slot for the formula in the instance's simulation subtable. Because the Shell's variables are larger structures that require more memory space, this invention uses by default subclasses of the Shell's parameter as attributes of bioObjects. However, alternative but equivalent subclasses of variables are also provided, allowing the user to delete the default parameter and replace it with the equivalent variable, allowing the user to then write any customized specific simulation (values provided by the simulator) or general (values provided by the inference engine) formula to model that particular variable. Differences between the Shell's parameters and variables include: a) parameters do not have slots for specific formulas but can receive values from generic formulas that apply to instances of a class; and b) a variable may have an initial value and two current values, the inferred and the simulated values, while a parameter have only one value that can be either inferred or simulated; c) the inferred value of a variable can be set to expire after a time interval and therefore a variable may have no value (it should not expire when the value is used by the simulator), while a parameter always has a value; c) variables can backward-chain to seek its values, while parameters cannot; and d) variables cannot be directly referenced by procedures. When using a different Shell with parameters and variables with different attributes and capabilities, some modifications may be necessary. Examples of default instances are shown in FICs.12 through 15 for variables such as model-block-output-var (1228) and model-block-var (1230), and for parameters such as: basal-density-par (1306), scaling-density-par (1308), density-entry-pvar (1310), scaled-entry-pvar (1312), input-rate-pvar (1314), density-pvar (1318), basal-conc-par (1322), conc-pvar (1324), contribution-pvar (1415), consum-rate-pvar (1417), binding-rate-pvar (1506), time-delay-par (1511), ph-par (1513), ph-factor-par (1515), and produc-rate-pvar (1523).

[0088] The details of a variable are shown in FIG.12 with the attribute slots provided to enter specific information to define and control each particular instance (1228, 1230), which may include an additional simulation subtable (1232). An example of why and how specific instances of the default parameters are to be substituted when needed for the larger variables is discussed here. The Velocity attribute (1505) of an instance (1501) of a subclass of bioEngine is an instance (1501) of a subclass of rate-pvar, which value is simulated according to a generic-simulation-formula that refers to such subclass (example in Table 1) and represents a mathematical model for all reactions of such type. The Velocity attribute is allowed more flexibility in this invention because there are not only many types of processes, depending on the participants in each case, but there are also many opinions about what is the most fitting equation to compute the Velocity of each type. In addition, there are two types of simulation provided by the system of this invention, one is semi-quantitative, using scaled (dimensionless) variables, and the other uses variables with absolute values. For the Velocity attribute, depending on the bioEngine subclass, different subclasses of rate-pvar are used, such as: catalytic-rate-pvar, binding-rate-pvar, dissociation-rate-pvar, modification-rate-pvar, or transfer-rate-pvar, with such scaled-values based formulas. However, the modeler can substitute those with an appropriate instance of an alternative set of more specific subclasses of velocity-pvar with more specific absolute-value based formulas. Each subclass of velocity-pvar, such as: enzyme-velocity, receptor-velocity, complex-formation-velocity, complex-dissoc-velocity, ion-transport-velocity, conform-change-velocity, and translocation-velocity, has more specific subclasses, and each of those subclasses has its associated more specific absolute-value based simulation formula. The nomenclature used for naming each subclass indicates all the participating components and the reaction type. For example, 1E.2S. 1I.ORD.velocity is meant for a bioengine.1E.2S.1I, which means that its icon has at the top four specific stubs connected to one enzyme.r, two substrate.r and one inhibitor.r, and the formula for that instance reflects the fact that the reaction happens in an ordered fashion, that is substratel.r binds before substrate2.r, and the inhibitor is a competitive inhibitor of the first substrate. Another alternative is provided for the modeler to write the simulation formula that most appropriately fits the system to be modeled, substituting with an instance of the class velocity-var described above.

[0089] The system of this invention utilizes the history keeping capability for the values of relevant parameters and variables, as specified in their listed definitions, to graph the time-line or to reason about those historic values. The user can decide to display those values in charts or other display forms. The historic values, as well as their maximum, minimum, average values. or rates of change over time can be not only be displayed, but also used by the user in different ways, such as storing them in external files or databases, allowing the values to be further analyzed.

• **Arrays and Lists**

[0090] Arrays and lists are structures that hold a number of item of values in an ordered way. The subclasses of arrays and lists have all the characteristics of their parent classes, as defined in the Shell, and additional attributes within this invention specified in their definitions. Arrays are indexed, and therefore each of its elements can be directly referred to by referring to its index, while lists may be indexed or not. The length of arrays have to be predetermined, while the length of lists increase automatically, as more elements are added. The current implementation of this invention uses arrays to permanently store the transient values of its elements by setting the Initial-values attribute of an array equal to the desired values to be stored, separated by commas, which may be the current values of the elements of that particular array at any point in time, or the elements of an auxiliary list, as described for queries. One of the classes of arrays used is query-array, a subclass of symbol-array, which instances are created by the query initialization procedures. For example, one of the uses of a set of query arrays is for storage in each of all the instances of each class of molecular components in the Virtual Model. The instances of references-array, a subclass of text-array, refer to sets of instances of the class reference-block and may be attributes of instances of bioEntities, bioReservoirs or bioProcesses, or bioModels. Furthermore, instances of the Shell's class float-array can be used to store all the values of a desired parameter or variable during a monitoring session or a simulation run, which can then be stored to a file or passed to another external program or device for further analysis or processing.

[0091] Bioitem-list is a subclass of the Shell's class item-list with no additional attributes. The different subclasses of bioitem-list are restricted to instances of particular classes of objects, such as experiment-selections, bioReservoirs, and bioProcesses. This invention uses lists in various ways, including: a) as transient auxiliary structures to process information at run-time, not visible to the users, such as instances of query-list, a subclass of symbol-list, are used as auxiliary structures in the creation of query-arrays, and instances of scroll-text-list, a subclass of text-list, are used as auxiliary structures in the creation of scroll-areas; and b) as auxiliary structures with their icons upon a workspace, which in addition to be used to process information at run-time they are available to the user for inspection, such as those shown in FIGs. 31 through 35 listing upstream and downstream bioReservoirs and bioProcesses, in reference to the bioObjects from which they where requested. Clicking on those icons by the user causes the display of the elements of the list, providing direct access to the objects listed and to the attribute table of the objects listed.

BioTools

• **Class model-block**

[0092] The **class model-block**, a subclass of the class bioTool, may have as many subclasses as needed to represent the different mathematical models needed by the domain modeler, and some representative subclasses are shown in FIG.19 as examples. Some examples of generic simulation formulas used to compute the value of the outputs for those subclasses are shown in Table 6. Also included is a generic-model-block, to be further defined by the modeler, which is a place holder for new parameters and/or variables to be added, and for specific general or simulated formulas for those newly added variables. The additional attributes for the class model-block includes Output.1 and each subclass have additional specific attributes which are simple attributes, parameters or variables, and which vary in number and type. A model-block is an iconic object, preferentially used upon the subworkspace of a model-box, which encapsulates the components of a mathematical model, as defined by its attributes, where the value of one or more of those attributes, the outputs, are dependent on the values of inputs, which may be constant or may changed at run time. The inputs may be have any type of origins, including: a) one or more other attributes of the model-block; b) one or more attributes of other objects in the Virtual Model; c) one or more values originating from any external source interfaced with the system, such as external sensors, databases, or external simulators; d) any auxiliary independent constant, parameter, or variable; e) any timer or meter: or f) any combination of the above.

[0093] As shown in FIG.12, clicking on an instance of model-block (1222) displays its menu (1223). The "disable" option (1224) is used when a set of alternative model-blocks is configured for a given bioPool to disable those not currently in use. The "table" option (1225) displays the attributes (1226) and provides access to its variables. Two classes of variables defined to be used as attributed of model-blocks: model-block-var and model-block-output-var differ in the default settings: only the former has by default a simulation subtable and only the latter keeps history of its value, while both have an initial value and indefinite validity interval, so that they always have a value, and therefore can be used by the simulator. Clicking on the slot (1227) of the attribute Output.1 displays its subtable, the attribute table of an instance of model-block-output-var (1228). Although the predefined model blocks classes have associated generic formulas to compute the value of their output.1, the modeler may want to override it by requesting, by clicking on the slot of its simulation details attribute, the addition of a simulation subtable. The subtable of the attribute Input. 1 (1229) shows the table of an instance of model-block-var (1230), and clicking on the slot of the attribute Simulation-details (1231) displays its simulation subtable (1232) with additional attributes, among them one (1233) to hold the simulation formula to be written by the modeler. The particular subclass selected in this example, a proportional.f, is defined with two additional attributes given by simple float attributes with default values of 1.0 for Gain (1234) and 0.0 for Bias (1235), which represent what their names indicate.

• **Class inference-block**

[0094] The **class inference-block**, a subclass of the class bioTool, may have as many subclasses as needed to represent the different inference models needed by the domain modeler, and some representative subclasses are shown (Table 8) as examples. Some examples of generic simulation formulas used to compute the value of the outputs for those subclasses are shown in Table 7. The additional attributes for the class inference-block includes Outcome, given by an outcome-par, and each subclass have additional specific attributes which are simple attributes, parameters or variables, and which vary in number and type. An inference-block is an object, preferentially used as an attribute of bioObjects, such as time-compartments, which represents an inference model, as defined by the methods providing the value of its Outcome, which is dependent on the values of some inputs. The inputs may be have any type of origins, including: a) one or more other attributes of the inference -block; b) one or more attributes of other objects in the Virtual Model; c) one or more values originating from any external source interfaced with the system, such as external sensors, databases, or external simulators; d) any auxiliary independent constant, parameter, or variable; e) any timer or meter; or f) any combination of the above.

• **Class button**

[0095] The **class button** is a subclass of bioTool, which additional attributes include Toggle-state. A subclass of button is action-butto), which additional attribute, action-proc-name, holds the name of the specific procedure invoked when that button is selected by the user. Clicking an action-button triggers its "select" option which invokes the button-handler-callback, which invokes the procedure defined as attribute of each subclass, such as the hide-sup-workspace-callback for a HIDE button. Action-buttons play a supportive role in this invention by controlling the display, and in some cases the activation, of workspaces, allowing the interactive navigation through the entire Virtual Model. Other sub-classes of button, named tracers, initiate important interactive tasks to create and display pathways, lists, or graphs from a point of view within the Virtual Model relative to the bioReservoir or bioProcess from which they are selected.

[0096] The **class change-mode-button** is a subclass of action-button with the additional attribute Mode, which upon selection starts the change-mode-callback. A set of these buttons on a panel, with the Mode and Label attributes of each button of the set equal to each of the possible values of the user-mode attribute of the window, allows the user to interactively change from one user mode to another, for the particular window displaying the panel. The optional behaviors of many types of objects in this system are user mode specific, as defined within the Class-restrictions attribute of the object-definitions of the corresponding classes, such as the actions invoked when selecting their icons, the options that appear on their menus, or the visible attributes.

[0097] The following classes are subclasses of the class button with no additional attributes, but have subclasses with specific methods associated with them: a) the **class path-tracer**'s "show" option starts the path-tracer-handler-proc, which calls the configure-tracer-proc and one of three optional procedures depending on the class of the path-tracer, as described in later sections related to bioProcesses, Navigation Panel, Simulation Panel, and Experiment Panel, where they play a support role by controlling the dynamic creation and display at run-time of interactive pathways of either bioReservoirs or bioProcesses or both; b) the **class lists-tracer**'s "show" option starts the list-tracer-handler-proc, which calls the configure-tracer-proc and one of three optional procedures depending on the class of the lists-tracer, as described in later sections related to bioReservoirs and bioProcesses, where they play a support role by controlling the dynamic creation and display at run-time of lists of either bioReservoirs or bioProcesses that are either upstream or downstream of the bioReservoir or bioProcess where the lists-tracer is located; and c) the **class graph-tracer**'s "show-plot" option starts the graph-tracer-handler-proc, which calls the configure-graph-tracer-proc and one of three optional procedures, depending on the class of the graph-tracer as described in later sections related to bioReservoirs and bioProcess, where they play a support role by controlling the dynamic creation and display at run-time of graphs that plot the values over time of key variables or parameters of those bioObjects.

• **Other subclasses of bioTool** include:

[0098]

a) instances of the **class reference-block** are used to represent each reference to published information that serves as the source for building the different parts of the models. Selecting a reference-block displays information about the sources, such as the short-reference, authors, title, journal, institution and abstract; b) instances of the **class bioObject-title** are used to label subworkspaces, such as those of bioEntities, bioEntity-notes, subModels. A set of rules automatically sets the text of those titles, based on the label of the object superior to such subworkspaces; and c) the classes **reference-message** and **sequence-display** are subclasses of the Shell's class message used for display of text in the References Scroll and the Mol.Sequence, described below.

Group of Operating/Display Tools

[0099] The classes query-panel and entry-panel are both subclasses of the class uil-tailored-dialog defined in G2 from which they inherit several attributes used for run-time processing. The different types of query-panels and entry-panels are designed as partially built master structures which are dynamically cloned and completed at run-time, upon requests from the user. The dynamically created Entry Panels (FIGs. 32, 33 and 35) are used to interactively set-up and start constrained navigations, simulations, and experiments, and for selection of monitoring/control components. The dynamically created Query Panels (FIG.30) are used to interactively perform predefined complex queries, based on functional modular structure, relative position within the pathways, function (roles as bioReactants), or location in subcellular compartments.

[0100] Scroll-areas are structures provided by the Shell. In this invention Scroll-areas are created dynamically and used extensively (FIGs.30 through 35) to list and provide direct access named instances of different types of bioObjects. For example, a scroll-area is created upon request from an Experiment Panel (FIG.35) by selecting the BIORESERVOIRS button (3506), listing in a scrollable fashion all named instances of bioReservoirs in the Virtual Model and providing direct access to them. The scroll-messages that form the scrollable components of the scroll-area, and which refer to the objects of interest, can be ordered alphabetically or in other ways.

Connections and bioPosts

[0101] **Connections** are used to connect visualObjects (FIG. 7) upon a workspace and are established graphically, by the union of the stubs attached to the bioObjects involved or by extending a stub of a bioObject and attaching it directly to another bioObject, if the latter allows manual connections. Several domain-specific connections are subclasses of the Shell's class connection, and represent different abstract concepts such as inputs, outputs, links, relationships, or information passed between objects. Stubs are handles (716, 718, 720) on the icons of objects that are clicked and dragged to create the connections (717, 719). The definitions of the subclasses of bioEntity, bioPool, bioEngine, bioReactant, bioProduct and bioPost (703, 704, 711, 712) prescribe the class of connections that are allowed at each port of their icons, as defined in their stubs slot. Direction of flow may also be specified in such definitions. The stubs are inherited, and those stubs that are not used can be graphically removed.

[0102] The definitions of connection subclasses include the descriptions of the **Cross section pattern**, that provide each with characteristic pattern and color to facilitate visualization and modeling tasks, and a predefined **Stub length**. Each of those classes is specific for connecting particular combinations of objects (FIGs. 6 and 7). Stubs of different classes or subclasses are not allowed to connect to each other, to prevent the modeler from making erroneous connections. Connections comprise the following major classes:

- the **class p-connection** represents a bioProcess input to a bioReservoir, and more specifically a bioProduct's input to a bioPool, in terms of both unidirectional material flow and data flow. Several stubs for p-connections are defined at the top (716) of the class bioPool, which may graphically connected (717) to the one stub defined at the bottom of each biopool-p-post (712). Also several stubs for p-connections are defined at the bottom (720) of the class BioEngine, which may graphically connected to the one stub defined at the top of each bioProduct.

- the **class r-connection** represents a bioReservoir's input to a bioProcess through a bioReactant, in terms of both unidirectional material flow and bidirectional data flow. An r-connection may connect (719) a biopool-r-post to the bottom of a bioPool, or may connect a bioReactant to the top of a BioEngine. Several stubs for r-connections are defined at the bottom (718) of the class bioPool. In addition, the class r-connection has many subclasses which differ in their characteristic cross section color pattern, and each connects a different class of bioReactant to a matching stub of the same subclass defined at the top of a bioEngine. The definition of each subclass of bioReactant specifies within the stubs attribute the specific subclass of r-connection, and the definition of each subclass of bioEngine specifies within the stubs attribute a set of specific subclasses of r-connections and their locations. The purpose of this implementation is to prevent users from making the wrong connections, while directing the user to make the appropriate connections by matching the color pattern.
- the **class model-box-connection** has three subclasses, scaled-input-box-connection (721), input-box-connection (722), and output-box-connection (723), used (FIG.8) to graphically connect instances of scaled-input-model-box (811), input-model-box (816), and output-model-box (817), respectively, to predefined corresponding ports of a bioPool. The stubs for these specific connections are defined for each subclass of model-box, and for each subclass of bioPool at named ports, and these subclasses are color coded to guide the modeler to match the color pattern when making the connections;
- the **class link** defines the connections between the component bioEntities (1819, 1821) of other bioEntities (1817), representing the continuity of an ordered sequence of structural components.
- the **class icon-wire** is a tool used to connect a bioRole-post (703, 707, and 711) to its bioRole-Object. The **class graph-link** is a tool used to connect either a bioEngine to various types of tracers (3105, 3110, 3117, 3426) or a bioPool to various types of tracers (3421). After the connection is made, these connections are made invisible by either a rule at run-time or by a procedure invoked at initialization time;

[0103] The **class BioPost** is a subclass of of the Shell's class connection-post. Connection-posts are like extensions of connections that permit to physically interrupt a graphic connection and continue it on another workspace, but without interrupting the flow. A teaching of this invention is the design, uses, and methods associated with the different subclasses of bioPost, each referring to an iconic structure (703 or 711) that may be distantly connected to another bioPost of a complementary subclass (704 or 712, respectively). BioPosts are components located on the subworkspaces of bioReservoirs and bioProcesses and, since they are located upon different subworkspaces, they are distantly connected by receiving the same name. These structures integrate bioReservoirs and bioProcesses, wherever they are located. There are several restrictions built into the system of this invention to prevent users from making the wrong connections, or to warn them that wrong connections have been made, such as when a bioReactant-post is given the same name as a bioPost that is not a biopool-r-post, or when a bioproduct-post is given the same name as a bioPost that is not a biopool-p-post. The various subclasses of bioPost will be further described below, within the overall description of bioReservoirs and bioProcesses.

BioObjects

[0104] The class **bioObject** is a subclass of the Shell's class object, with various subclasses and various restrictions when in different modes. BioObject comprises all the classes of knowledge structures used to represent the biochemical systems characteristic of this domain. These object classes are characterized by their defined attributes, which can be: a) simple attributes of various types, such as text, symbols. integers, floats or truth values; or b) other previously defined objects, including parameters and variables of any of those types, or lists and arrays of those types of elements. The values for the parameters and variables can be provided by any of a variety of sources, such as the user-inputs, interfaced on-line sensors, or dynamically by the inference engine (for instance, through specific or generic formulas, rules or procedures), or a simulator (for instance, through specific or generic simulation formulas or simulation procedures), reflecting the changing values of the attributes defined within the knowledge structures and in the context selected by the user.

[0105] All bioObjects have at least three common attributes: a) **Names**, a Shell-defined attribute that binds one or more unique symbols to an object; b) **Label**, a text attribute to identify bioObjects that may not have a unique name; and c) **Description**, a text string for any additional information. While names are required in the current implementation for some bioObjects to be integrated into the pathways, such as bioReservoirs, bioProcesses, and bioPosts, labels are preferred for identification in other cases, because they require less memory than symbols, do not have to be unique, their syntax is less restrictive, and offer more attractive options for display.

[0106] Among the major bioObject's subclasses, which further subclasses are described in later sections, are:

class **bioNode-Object**: inherits all attributes from bioObject and have no specific attributes, and can only have subclasses, not instances However, there are methods that refer to this class, and therefore, like with many other classes throughout this system, the differentiation between classes may not appear in their definitions but rather on their behavior, through methods that applied to them.

class **bioView-Object**: inherits all attributes from bioObject, and can only have subclasses, not instances. A option **"details"** defined for this class upon selection by the user starts the procedure go-to-view-proc, which takes among its arguments the instance from which it was invoked and the current window and, depending on the value of the instance's **Toggle-state** attribute of hide or show, it causes the instance's subworkspace to be displayed or hidden, it toggles such value, which causes the instance's icon to be configured to its pressed or depressed appearance. New attributes defined for this class comprise:

References, defined for several subclasses of bioObject, is an optional reference to the sources of information and data used to construct and configure that instance of any subclass of bioObject for which it is defined. This attribute may be defined as: a) a simple text attribute, which the user may consult to perform independent searches or using this invention search facilities, such as the Master-Reference-Panel: or b) a reference-array in combination with the option "show-references", which when selected by the user starts the show-references-of-array-proc which displays dynamically created copies of the instances of reference-block referred to by the values of that array.

[0107] **Warnings** is an optional simple text attribute which value is set by the modeler to warn the user about any abnormalities or data included, such as, for a bioReservoir, when that particular pool of molecules have not yet been observed experimentally, but can be assumed to exist by analogy to similar systems. This attribute is used in conjunction with a design (724, 843) in the icons defined as a color-region, called flag-color, that is changed to yellow to make the user aware of the existence of warnings. Those changes are automatically executed by procedures invoked during initialization, or at run-time by a set of rules.

[0108] **Toggle-state** is an attribute also defined for objects which icons' appearance change upon selection by the user, depending on the value of this attribute which switches between show and hide when the user clicks on the icon.

BioEntities

[0109] A bioEntity is an iconic knowledge structure that represents any biochemical entity or their components, at different levels of structural complexity. Each bioEntity may have information encoded in various ways, including: a) information and data stored in its table of attributes, and b) a subworkspace upon which the components that represent its functional modular structure are visually defined. As shown in FIG.30, selecting from a bioEntity icon's (3004) menu (3005) the "details" option (3006) displays its structure (3007), which may have several layers of detail (3008 - 3022). In the current implementation, functional components represent structural components relevant for the bioEntity's function and regulation that allow the user to visualize the structural composition. The iconic composition is also used in the inference for processing of queries that involve the structure, which refer to bioEntities that have certain components in their subworkspace. Each functional component of a bioEntity belongs itself to a bioEntity class, either molecule or bioEntity-component or any of its hierarchy of subclasses.

The **class bioEntity**, a subclass of bioObject, has numerous subclasses organized in a hierarchical structure with different levels, according to the way a biochemist would classify different molecules by taking into consideration their commonalities in chemical composition in the upper levels of the hierarchy, such as protein, nucleic acid, or lipid, followed by a reference to their function, such as, within the class protein, active-polypeptide, receptor, or enzyme. Additional attributes include **References**, described above. There are two major subclasses of bioEntity:

The **class simple-bioEntity** includes all the subclasses with simple structures that are not further visually described in the current implementation, and they do not generally have subworkspaces with additional components. This class has two subclasses, protein-site, and protein-modified-group, each with its additional subclasses: Additional attributes include:

Id is a simple text attribute used to identify the instance instead of using a unique name;

Position is a simple text attribute to indicate the position of that component within the molecule when applicable, such as the position of an aminoacid in the sequence of a protein, and is usually displayed; and

Superior-bioentity is a simple indexed attribute that holds the name of the first superior bioEntity in the workspace hierarchy that has a name, if any, a value that is computed by the program within the query reasoning and, as explained later, this mechanism is provided to avoid repetition of structures that are shared by various instances of the same family or of other families that share regions of homology.

[0110] The **class complex-bioEntity** includes all those subclasses that can be further described by their components in their subworkspaces. Additional not previously described attributes include:

Master-bioentity is a simple indexed attribute that may hold the name of a bioEntity that is displayed when the user selects the "details" option. This value is set by the program upon cloning an existing named bioEntity using the "create-local" option. BioEntities with no names which Master-bioentity names another bioEntity, the master bioEntity, can be considered as only a copy or pointer to the subworkspace, or **Master-details**, of the master bioEntity, but it can also hold additional information in its table of attributes or in its own subworkspace, or **Local-details**.

[0111] The major subclasses of complex-bioEntity, defined to deal with multiple levels of biochemical structural complexity. include:

the **class bioEntity-component** and its subclasses represent he lower-level molecular functional components such as: dna-component and its subclasses, such as dna-domain, promoter, operator, and so on: protein-component and its subclasses, such as protein-domain, protein-motif, and so on: and membrane and its subclasses. The subclasses of this class may have 4 stubs of the class link used to link different bioEntity-components, allowing for branching, and those stubs that are not used can be dragged away. Additional attributes include: **Superior-bioentity** and **Position**, defined above; and Size.
the **class molecule** and its successive level of subclasses such as those of **protein, nucleic-acid**, and so on, represent the next level of complexity. For example, as shown in FIG.30, a protein may have a set of domains or motifs, or a combination of both, and it may have in addition one or more protein-sites, or protein-modified-groups, connected to the domains and or motifs at approximate locations, with the more accurate position displayed in the Position attribute. A protein-domain may also have in its SW a set of smaller domains or motifs, or a combination of both, and it may have in addition one or more protein-sites, or protein-modified-groups, including copies of those connected to the icon of the domain.
the **class heter-mol-complex** and its subclasses, such as nucleosome or tf-dna-complex, represent a higher level of complexity. A heter-mol-complex or a complex-molecule may be composed of other molecules.
the **class membrane** and its subclasses, are used to represent interactions of several types of cellular membranes with other molecules;

[0112] Additional information about an instance of bioEntity can be stored in its table of attributes. The inherited attributes shown are successively added through the hierarchy of classes such as molecule, protein, and enzyme, and include:

Synonyms is a simple text attribute to list the different names by which a molecule or complex may be known;
In-species is a simple text attribute to list the species or common name of living organisms in which that molecule or complex has been found, such as all, human, mouse, Drosophila, C. elegans. S. cerevisiae, and so on;
In-tissues is a simple text attribute to list the tissues in which that molecule or complex has been found.
InCells is a simple text attribute to list the cell types in which that molecule or complex has been found.
InOrganelles is a simple text attribute to list the organelles within a cell in which that molecule or complex has been found, and it may take values such as plasma-membrane, cytosol, nucleus, and so on.
Cas-number is a simple text attribute to indicate the CAS number assigned to that molecule. which is indexed to allow for faster searches and has a default value to indicate the preferred format;
Mol-weight is a simple integer attribute to indicate the molecular weight of a single molecule or complex, preferably expressed in Daltons;
Isoelectric-point is a simple float attribute to indicate the isoelectric point of the molecule or complex;
Sequence is a simple text attribute to contain information about the sequence of a molecule, if relevant;
Substrates-info is a simple text attribute to contain information about the known specific substrates or groups targeted by the enzyme;
Inhibitors-info is a simple text attribute to contain information about the known specific inhibitors of the enzyme;
Ligands-info is a simple text attribute to contain optional information about any other ligand that may bind to the enzyme;

[0113] The **class bioEntity-notes**, a subclass of bioTool, comprise auxiliary structures located the subworkspace of a bioEntity, that serve as iconic containers for different types of specific information about the bioEntity, which can be displayed by selecting the "details" option.

BioReservoirs

• Class bioReservoir

[0114] As shown in FIGs.8 and 12, a bioReservoir (801, 1201) is an iconic object that represents an imaginary container of a population of similar molecules or molecular complexes, represented by a bioPool (805, 1301). A bioReservoir has several operational functions and encapsulates several forms of knowledge (FIGs. 8, 12, 13). A bioReservoir's menu (802, 1202) provides access to a variety of tasks to be discussed in sections below. Here we will define some of the components involved. The "table" option (1203) displays its attributes (1204), which holds values set by the modeler that, in addition to provide useful quantitative or qualitative information to the user, may be used by the program to set the initial conditions before a simulation is run. The values of some of the attributes of a bioReservoir characterize the system and are stored as part of the permanent database. The values of other parameters and variables (1305 - 1324) pertaining to a bioReservoir which are computed when a simulation is run, but which do not usually require to be set by the modeler, are hidden for convenience as attributes of the bioPool encapsulated in each bioReservoir, but they could as well be implemented as attributes of the bioReservoir.

[0115] The **class bioReservoir** is a subclass of bioView-Object and its subclasses include: bound-mol-reservoir (801), sol-mol-reservoir (1201), cell-reservoir, exp-cell-reservoir, genet-mol-Reservoir, some with further subclasses. In FIG.12 is shown a table of attributes (1204) of an instance not yet configured, shown here with its default values. The unlikely default value of 9.9e-99 given to some of those attributes, which provides a) a signal for the user that an adequate value has not been entered yet by the modeler, and b) a branching criteria for the inference engine. Attributes newly defined for this class and its subclasses not previously described include:

Compartment (1205) is an simple attribute optionally set by the modeler to represent the physical boundaries of the encapsulated bioPool;
Status (1206) is a symbolic attribute which value is set and used at run time by the simulation and other procedures, as an aid in the activation of pathways and in the creation of interactive pathway displays. It can take any of the values specified in the definition table and it is not visible to the modeler or other users.
Ref-bioentity (1207) is an optional attribute optionally set by the modeler to point to the bioEntity instance that describes the structure of a unit representative of those populating that bioReservoir. This value is used by the program when the user selects the "bioEntity" option from the menus of either the bioReservoir itself or any of the bioReactants and bioProducts distantly connected to its bioPool, and that option appears on those menus only when the value of this attribute is not the default value none;
Master-bioreservoir (1208) is set only by the program when copies of named bioReservoirs are made, such as those used in the creation of interactive pathway displays, and is used by the program to display the subworkspace of the bioReservoir referred to by the value of this attribute, when the "master-details" option is selected by the user. This attribute is not visible to the modeler or other users.
Decay-rate-factor (1209) is a parameter which value may be entered by the modeler or can be modified at run time by rules or procedures. Its default value is 2.0e-4, a value small enough not to cause dramatic changes, but in line with some experimental observations in the domain of this invention, but it should preferably be modified by the modeler to better reflect individual cases;
If-scaling-amount (1210) is a simple attribute set by the modeler to indicate to the program the value to be used to interconvert between the absolute-valued and the scaled-valued variables or parameters of bioPools and their connected bioReactants. The complex methods used for this conversions differ, depending on the classes of bioPools and bioReactants involved. Its default value of 100 is used if not modified by the modeler, if the value of the if-scaling-bioreservoir attribute is none;
If-scaling-bioreservoir (1211) is a simple attribute which value may optionally be set by the modeler to indicate to the program that the value of the if-scaling-amount should be taken from the bioReservoir named by this attribute. This value is only required or used when the value of the if-scaling-amount attribute is the default, otherwise, the later value is used and this attribute is ignored. This and the previous attribute are not visible in General Mode;
Scaled-basal-amount (1212) is a float simple attribute set be the modeler that represents the scaled value of the amount of units in the encapsulated bioPool under normal or basal conditions, equivalent to a fraction of the maximum amount that this bioPool can reach under optimal physiological conditions. Since in many occasions in the domain of this invention neither the basal nor the maximum amounts can be measured or known with certainty, this value represents in such occasions the best estimated guess, and it is an important semi-quantitative knowledge component. The default is the unlikely value of 9.9e-99.
Physiol-abundance (1213) is a symbolic attribute which value is set by the modeler to indicate the physiological level of abundance of the entity in the compartment represented by that bioReservoir. It is a symbolic replacement of the previous attribute when the 9.9e-99 has not been modified by the modeler. The values allowed for this attribute in this invention are: highest, abundant, steady-state, low-induced or only induced, which currently correspond to the scaled values of 1.0,0.9,0.5,0.1, and 1.0e-9, respectively. However, this default scale can be changed to different values, and additional symbols and values can be added to the scale. As and alternative, fuzzy-sets can be defined for the values of this attribute, and fuzzy-logic can then be applied to those values. The reasoning for using the values of either the previous or this attribute, in conjunction with one or more of the following attributes, are discussed in more detail under the Simulation Mode heading.
Normal-basal-concentration (1214) is a float simple attribute set be the modeler that represents the physiological average value of the concentration of the entity represented by a sol-mol-reservoir;
Normal-basal-density (instead of 1214) is a float simple attribute set be the modeler that represents the average value of the density of the entity represented by a bound-mol-reservoir.
Physiol-max-concentration (1215) is a float simple attribute set be the modeler that represents the physiological

maximum value of the concentration of the entity represented by a sol-mol-reservoir;
Physiol-max-density (instead of 1215) is a float simple attribute set be the modeler that represents the physiological maximum value of the density of the entity represented by a bound-mol-reservoir.

[0116] Selecting the "details" option (803) displays the subworkspace (1217) which contains the components that characterize a bioReservoir. The classes of those components that are using in the modeling process are defined below, while other auxiliary structures are described in later sections. The descriptive qualitative, structural, and functional information of a bioEntity is kept in this invention separated from the configurable additional quantitative information that characterizes each bioReservoir, allowing the Ref-bioentity (1207) of several bioReservoirs, representing populations of the same type of entity in different locations or points in time, to point to the same bioEntity to reduce the size of the Virtual Model.

• **Class bioPool**

[0117] The class bioPool is a subclass of bionode-object and comprises several subclasses, such as sol-mol-pool and bound-mol-pool. A bioPool (805, 1301) is defined with a characteristic icon with two sets of stubs: a set of p-connection stubs at the top of the icon, and a set of r-connection stubs the bottom of the icon. Each p-connection stub is to be connected to a biopool-p-post (806) and each r-connection stub is to be connected to a biopool-r-post (818, 823, 828, 833) on the subworkspace (804) of the bioReservoir (801), to establish distant connections to bioProducts (808) and bioReactants (820, 825, 830, 835), respectively, and allow bidirectional flow of data and control between them. As shown in FIG.13. every bioPool (1301) has a menu (1302), and selecting the "table" option (1303) displays its attributes (1304). Several of the attributes of a bioPool are variables or parameters (1305 - 1324), which include a density-related set and a concentration-related set, which values (with the exception of the basal quantities which are preferably set by the modeler if known but can be computed from the set values of other attributes of the bioReservoir, or their default values) are inferred or simulated. The units of all the parameters and variables of all bioPools connected to the same bioProcess have to be appropriately matched. Newly defined attributes of bioPool comprise:

- **Basal-Density** (1305) is given by a basal-density-par (1306). The subclass sol-mol-pool has additionally a **Basal-Concentration** (1321) given by a basal-conc-par (1322). The values of either of these attributes may be set by the modeler to indicate: a) the normal physiological steady state density or concentration, or any other appropriated measured or assumed value, for physiological molecules or complexes, or b) 0.0 for pharmacological or other molecules that are added to the system from an external environment. This value is required for any bioReservoir that participates in a simulation. and if its default value has not been overridden by the user with a new value, the program sequentially pursues other sources of the Basal-Density value, as determined by a generic basal-density-procedure which is called during the activation process of a simulation, only for those bioReservoirs that have been activated for a simulation;
- **Scaling-Density** (1307) is given by a scaling-density-par (1308) which value is inferred by a procedure invoked at initialization. This value is used by simulation formulas to interconvert scaled values and absolute values.
- **Density-Entry** (1309) is given by a density-entry-pvar (1310). It takes the value of a user-input at a time, or different times intervals, during simulation as defined by the user in an input-panel during the simulation set-up process. It has a default value of 0.0, and a new value may be inferred and set by a procedure called when the simulation is started, and after the new inferred value has been propagated it reverts back to 0.0. This value is an argument to the Accumulation, where it is summed to other inputs and outputs;
- **Scaled-Entry** (1311) is given by a scaled-entry-pvar (1312), and it is the scaled equivalent of the Density-Entry, with its value set and used in a way similar to that described above;
- **Input-Rate** (1313) is given by a input-rate-pvar (1314), which value is given by a simulation formula that sums all the visually defined inputs, including: a) the current values of the Production-Rate of all the bioProducts connected to the biopool-p-posts of the bioPool, if any; and b) the input modeled by means of a model-block encapsulated in any connected input-model-box (or scaled-input-model-box), if any;
- **Output-Rate** (1315) is similarly given by a output-rate-pvar, which value is given by a simulation formula that sums all the visually defined outputs, including: a) the current values of the Consumption-Rate of the consuming bioReactants connected to the biopool-r-posts of the bioPool, if any; and b) the output modeled by means of a model-block encapsulated in any connected output-model-box. if any;
- **Accumulation** (1316) is given by an accumulation-pvar, a time variant and continuous state variable which represents a quantity in the classical systems dynamics sense which, given an initial value here set to be equal to the basal-amount (or scaled-basal-amount), integrates the input-rate, the output-rate, the input entered by the user through an input-panel, if any, and the decay term, which is a function of the current value of the Concentration or Density as given by the decay-rate-factor of the bioReservoir (a decay rate constant in sec^(-1)) which represents

a variety of outputs not modeled visually or through a model-block, including the degradation and diffusion components).

- **Density** (1317) is given by a density-pvar (1318) which value represents the number of molecules, complexes, or cells per liter. or any other unit of volume, such as the volume of the reaction mixture in a reactor. The subclass sol-mol-pool has the additional attribute **Concentration** (1317) given by a scaling-density-par (1318), which value in M (Molar) represents moles per liter. In the implementation used to illustrate this invention, simulations operate by default based on the Density of the bioPools involved. In the case of sol-mol-pool, where the quantities used by scientists may be most commonly given as Concentrations, the second set of concentration related variables is defined to facilitate user-input, but those quantities are preferably transformed into density-related variables by the program, using Avogadro's number (6.023e23 molecules per mol), before integration with other quantities. However, several other types of quantities could be used as well, with conversions to other units of measure, such as activity (X-units times Y-unit), amount (X-units), or others types of densities (such as X-units per Y-unit). The same concepts and the same type of formulas, rules and procedures, apply to other-quantities, and a variety of units may also be allowed for entries from the user or other external sources, by providing lookup-tables or any other standard methods for automatically converting any of those units to the desired target units. Hence, the modeler has the option to design and define a system totally or partially based on a different quantity measurement, as long as proper care is taken to maintain consistency throughout the system. In those cases, the quantities entered by the modeler in the attribute slots of the bioReservoir that refer to physiological levels would also have to reflect those different sets of units.

[0118] **Scaled-Amount** (1319) is given by a scaling-density-par (1320), which value represents an scaled, dimensionless, alternative to Density, Concentration, or any other absolute-valued quantity. This attribute is used in the default scaled reasoning, which is the preferred method when dealing with biological systems, when absolute quantitative data is incomplete, and much of the data available is from relative measurements. The Scaled-Amount can be converted into a Density, and viceversa, by using the value of the Scaling-Density of the bioPool.

• Class bioPool-post

[0119] The **class bioPool-post** is a subclass of bioPost and has two subclasses: biopool-p-post (712, 806), which is to be connected (717) to an input p-connection at the top of a bioPool's icon, and its name is to be given to one bioproduct-post (711, 807) to establish a distant connection; and biopool-r-post (704. 708, 818, 823, 828, 833) which is to be connected to an output r-connection at the bottom of a bioPool's icon, and its name is to be given to one bioReactant-post (703, 707, 819. 824, 829, 834) to establish a distant connection. BioPool-posts are located upon the subworkspace of a bioReservoir and connected to a bioPool, which function in combination with bioRole-posts is to connect the bioPool to one or more bioEngines. The additional attributes for this classes include:

Id is a simple text attribute used as a unique identifier, instead of the unique name, in some types of processing, such as in the alternative simulation procedures;
Ref-bioprocess is a pointer to the bioProcess instance that encapsulates the bioProduct or bioReactant to which the biopool-p-post or biopool-r-post, respectively, are distantly connected. The value is interactively set by selecting the "set-refs" option of the bioproduct-post or bioReactant-post to which it is distantly connected, and it is used by the program for interactive navigation through bioReservoirs and bioProcesses.

• Class model-box

[0120] As shown in FIG.12, a model-box is an iconic object (1218, 1236, 1237) located upon the subworkspace (1217) of a bioReservoir (1201) and connected to its bioPool, which function is to hold and connect the optional model-blocks (1222) on its subworkspace (1221) to the bioPool. The class model-box is a subclass of bioTool and has three subclasses, each playing a different role as their name indicate: scaled-input-model-box (1218), input-model-box (1236), and output-model-box (1237). The menu options available for all subclasses of model-box depend on the user mode. Clicking on a model-box (1218) in any of the user modes bypasses its menu (1219) and selects the "show-sw" option (1220), which causes its subworkspace (1221) to be displayed.

BioProcesses

• Class bioProcess

[0121] The class bioProcess is a subclass of bioView-Object and comprises the subclass cell-bioProcess. As shown in FIGs. 7, 14 and 15, a bioProcess is an iconic object (701) which subworkspace (702) contains components that visually represent the preconditions (bioReactants, 1408, 1409), the process engine (bioEngine, 1501) and the effects (bioProducts, 1518) of a process as connected objects with encapsulated variables and parameters that describe the process qualitatively and quantitatively. The subworkspace of bioProcesses is activatable, allowing to control the availability of its components by activating or deactivating the subworkspace. The many different types of bioProcesses are all instances of that class or its subclass, and differ mainly in the iconic components upon their subworkspace. The different types are also visually distinguished by different colors for the type-color region of their icon, which is changed programmatically to be the same color as the color of the type-color region of the bioEngine they encapsulate.

[0122] A bioProcess has several operational functions, encapsulates several forms of knowledge (FIGs. 7, 14, 15), and has a menu (1402) that provides access to a variety of tasks. Here we will define some of the components involved. The "table" option (1403) displays its attributes (1404) which holds various attributes, some of which have values that are set by the modeler and provide information for the user, and others are auxiliary attributes hidden from modelers and users which values are set and used only by the program at run-time. Most attributes specific for this class have been previously described for other classes. Newly described attributes include:

> **Master-bioprocess** (1405) is set by the program when copies of a named bioProcesses are made, such as those used in the creation of interactive pathway displays, and is used by the program to display the subworkspace of the bioProcess referred to by the value of this attribute, when the "master-details" option (visible only when the attribute has a non-default value) is selected by the user. This attribute is not visible to the modeler or other users.

[0123] The "details" option (1406) displays the subworkspace (1407) with all its visual components. The main components comprise a single bioEngine (1501) which represents the interactions between the inputs represented by the bioReactants (1408, 1419) where the bioProduct(s) (1518) are generated.

[0124] As described below (FIG.29), a cell-bioProcess may encapsulate bioReactants and bioProducts that represent pools of cells, in addition to encapsulating bioReactants and bioProducts that represent pools of chemicals. So it is possible to have bioProcesses where molecules interact with molecules, as well as cell-bioProcesses where molecules (2905) interact with cells (2904), or cells interact with cells. In addition, there may be pools of pairs of interacting-cells. Other type of bioProcesses represent translocation processes between location compartments, where the bioReactant represents a fraction of a given bioPool in a given compartment which is removed from that bioPool and transferred, with a time-lag if so desired, to the target bioPool which is connected to the bioProduct, and which represents a different pool the same entity in a different compartment. Additional types of bioProcesses represent transfer of cells from a bioPool (2901) representing one state of those cells to another bioPool (2911) representing a different state of those cells. Each of these types of bioProcesses encapsulate the appropriate subclasses of bioEngines. There are many other types of bioProcesses defined in this invention discussed with the modeling approaches below. Many more types of processes can be represented by composing additional types of bioProcesses by a modeler skilled in the art.

• Class bioEngine

[0125] The **class bioEngine** is a subclass of bionode-object and comprises a hierarchy of subclasses. A few examples of which are: amplifier-bioengine, binding-bioengine, lumped-bioengine and cell-bioEngine. Each bioEngine's icon (1501) has a number of stubs of different types of r-connections at the top, to be connected only to bioReactants, and a number of stubs of p-connections at the bottom, to be connected only to bioProducts (2004-2011). The number and class of defined specific r-connections specify the different classes of bioReactants to be connected, such as enzyme.r, substrate.r, inhibitor.r, receptor.r, agonist.r, antagonist.r or unit.r (FIG.20). The bioEngines of the class lumped-bioengine, including those with two or more enzymes, represent lumped models of a two or more reactions in series. The major subclasses of bioEngine are arranged according to the type of reaction or process they represent, indicated by their names. The next level of classes usually refers to the type of activity of the members of the class and is usually characterized by the class of the variable that defines its Velocity attribute. Further subclasses of each of those classes are defined according to the number and type of their defined stubs, which determines the number and classes of connected bioReactants, and further by color coding the icons to show further specialization, such as using different type-color for different types of enzymes. Further differentiation between particular instances can be generated by selecting as the value of their Velocity attribute an instance of different subclasses of velocity-pvar. The nomenclature used for each subclass refers to the defined bioReactants, such as: a) bioengine.E1.S2.I1 is an enzyme-bioengine that represents an enzymatic reaction and which icon has at the top four stubs: one for the enzyme.r, two for substrate1.r and substrate2.r, and one for an inhibitor.r; b) bioengine.R1.L1.An1 is a receptor-bioengine which icon has at the top three-stubs: one for the receptor.r, one for ligand r, and one for an antagonist r; c)bioengine.U2 is a complex-formation-bioengine (2011) that represents the formation of a complex from two previously independent bioEntities (which may themselves be complexes) and which icon has at the top two stubs: one for unitl.r and one for unit2.r; d) bioengine. C1.M1 is a complex-dissoc-bioengine that represents the dissociation of one complex induced by a mediator into two or more independent bioEntities (which may themselves be complexes) and which icon has at the top two stubs for complex.r and mediator.r; or e) bioengine.ch1.i1.L1 is a channel-bioengine (2007) which icon has at the top three stubs: one for channel.r, one for ion-Input.r, and one for ligand.r.

[0126] As shown in FIG.15, a bioEngine (1501) is located upon the subworkspace (1407) of a bioProcess and represents the action of any process, such as synthesis, modification, complex-formation, translocation, diffusion, degradation, and so on. Selecting the "table" option (1503) from the menu (1502) of a bioEngine displays its attributes (1504), in this case with the default values. Attributes not previously described include:

Velocity (1505) is an attribute specific for each subclass which value is an instance of any of the subclasses of rate-pvar, described above, which for the receptor-bioengine shown is a binding-rate-pvar (1506). The values for this dependent variable are provided by generic simulation formulas that model the rate of the interactions of the bioReactants connected to the bioEngine. Its arguments may be the Contributions of each of its bioReactants, when using the scaled set of variables and formulas, or the bioReactant's kinetic-coefficients and the Densities, Concentrations or other quantities of the bioPools connected to the bioEngine through the bioReactants. Its output is an argument for the Consumption-Rate and Production-Rate of the connected bioReactants and bioProducts, respectively. Therefore, this attribute may take as arguments either a set of scaled-valued variables and/or parameters, in which case this attribute would be also scaled-valued, or a set of absolute-valued variables and/or parameters, in which case this attribute would be also absolute-valued.

Rate-constant-sec (1507) is a simple float attribute which value may optionally be set by the modeler to provide the rate-constant in seconds for the overall process. This value is used, in combination with the Contribution, by the set of generic simulation formulas that use the scaled set of variables, which is currently the default mode of operation. The default value is used if no other value is set by the modeler.

Tau-coeff (1508) is by default a simple float attribute which value may optionally be set by the modeler to modify the rate-constant-sec by such factor, allowing for testing the effects of a modification of such parameter on the simulation of the system without having to change the value of such parameter which may have been obtained experimentally. This value is incorporated by default in the generic formulas provided, and if not configured by the modeler, multiplying by its default value of 1.0 has no effect on the system.

Bias (1509) is a simple float attribute which value may optionally be set by the modeler to modify by a constant amount the rate-constant-sec, for testing purposes, without having to change the value of the later attribute. This value is incorporated by default in the generic formulas provided, and if not configured by the modeler, adding its default value of 0.0 has no effect on the system.

Time-lag (1510) is an optional time-interval attribute that may be useful for certain instances of bioEngines, which value, given by a time-delay-par (1511), is optionally set by the modeler to indicate the period of time by which the forwarding of the computed output of the Velocity of a bioEngine is delayed. It represents a time delay, equivalent to holding an amount from a bioPool for the specified period of time, which is returned after that time has elapsed. An example of its use is when a bioReactant and a bioProduct of this process are both connected to the same bioPool, and the forwarding of the output value of the Velocity is delayed. The value of this attribute may also be modified at run time, as a result of dynamic events related to this or other processes, and may be simulated or inferred by means of modeler-defined formulas, rules, or procedures;

pH (1512) and **Temperature** (1516) are optional float attributes which values are given by a ph-par (1513) and a temperature-par and which, as their name indicate, represent the environmental conditions under which the process takes place. The values of these parameters may vary at run time, as a result of dynamic events related to this or other processes, and may be simulated or inferred by means of modeler-defined formulas, rules, or procedures;

pH-deviation-factor (1514) and **Temp-deviation-factor** (1517) are optional float attributes which values are given by a ph-factor-par (1515) and a temp-factor-par and which are used in conjunction with pH or Temperature, respectively, and represent a correction factor that either accelerates or deaccelerates the process when the values of either the pH or the temperature deviates from their defined default values. If used, the values of these parameters may remain constant or may vary for different values of the pH or temperature, as defined by a tabular function similar to a look-up table.

• **Class bioRole-Object**

[0127] The **class bioRole-Object** is a subclass of bioObject, and has two subclasses: bioReactant and bioProduct. A bioRole-Object is an iconic object that represents the role that units from a bioPool plays in a bioProcess, either as a different types of bioReactants, such as: enzyme.r, substrate.r, inhibitor.r, subunit.r, receptor.r, agonist.r, antagonist.

r, carrier.r, and so on, each representing the specific roles of the inputs that different bioReactants provide to the bioEngine, or as a bioProduct.

[0128] As shown in FIG.14, a bioReactant (1408, 1419) is connected by an input r-connection at the top of a bioEngine, and represents the material contribution from a bioPool to the bioEngine to which it is connected. To be operational, the bioReactant-post (703, 707) connected to a bioReactant must have the same name as one of the bioPool-r-posts (704, 708) connected at the bottom of a bioPool. The two bioReactants shown are representative examples of a pair of two interacting molecules or complexes with complementary roles or functions: a receptor and its ligand. Many other combinations of two, three or any other number of interacting participants are possible, to represent other types of processes, such as combinations of enzyme and substrate(s), units that form a complex, and so on. These combinations may also include antagonists or inhibitors, which may alternatively be represented as competing in a separate process, the latter being the preferred alternative in the default implementation using the scaled set of variables and simulation formulas.

[0129] The **class bioReactant** comprises a hierarchy of subclasses, such as: amplifier-bioreactant, source-bioreactant, leading-bioreactant, binding-bioreactant, inhibitor-bioreactant, single-bioreactant, cellReactant, cell-receptor, extracell-ligand, and their subclasses. There are several attributes common to all subclasses of bioReactant, while each subclass has one or two specific attributes. Every bioReactant (1408, 1419) has its menu (1409, 1420), and selecting its "table" option (1410, 1421) displays its attributes (1411, 1422), in this case showing the default values for their attributes. Not previously described attributes specific for any of the subclases include:

Stoichiometric-coeff (1412) is a simple float attribute of all subclasses of bioReactant and bioProduct, which constant value represents the characteristic stoichiometric coefficient of that participant in that particular bioProcess;

Alpha-coeff (1413) is a simple float attribute which value may optionally be set by the modeler to modify by a factor the Contribution, which value is computed by the program. This is of interest for testing the effects of a global modification of such variable on the simulation of the system. This value is incorporated by default in the generic formulas provided, and if not configured by the modeler, multiplying by its default value of 1.0 has no effect on the system. Alternatively, the modeler may delete the alpha-coeffpar and provide instead a simple float value

Contribution (1414) is an attribute of every subclass bioReactant which value is given by a contribution-pvar (1415), which value is dynamically computed while a simulation is running. The meaning and use of this attribute is a novel teaching of this invention, designed to meet some of the challenges posed by the domain of this invention, where both qualitative and quantitative knowledge is frequently scattered and incomplete. This attribute represents the Contribution of each bioReactant to determining the overall rate of the transformation in that bioProcess, as represented by the Velocity of its bioEngine. The contribution is a variable that can take values from 0.00 to 1.00, and represents a dimensionless scaled concentration (or other equivalent quantity variable). The scaling of the concentration is done at the bioReactant level, rather than at the bioPool level, to allow for additional flexibility. In this way, the absolute concentration can be scaled differently for each bioEngine, centering around the specific constant that characterized the molecular interactions between each specific pair of bioReactants. The type of constant may be different for different bioReactants representing the same BioPool depending on the role played by each bioEntity in a given bioEngine, such as a Km for each substrate, the Ki for an inhibitor, the Ks for ligands or complex-subunits. The default values for all types of contribution are equal to: a) 0.5 for normal steady-state physiological conditions (for that particular compartment); b) 0.0 for bioPools of inducible proteins or non-physiological molecules; or c)1 for constitutive and repressible proteins. The user can either directly override a relative value or leave the default values. A generic formula specific for each of the bioReactant classes determines how the value of each contribution class is computed.

Consumption-Rate (1416) is an attribute of every subclass of bioReactant which value is given by a dependent consum-rate-pvar (1417), which value is dynamically computed while a simulation is running, and indicates the rate of consumption of units of the connected bioPool in such bioProcess. Its arguments are the bioReactant's stoichiometric-coeff and the Velocity of the bioEngine, and its value is an argument for the Output-Rate of the connected bioPool. For those classes of bioReactants that are not consumed in a reaction, the value of this attribute remains 0.0. In this invention, an enzyme.r or an inhibitor.r are preferably not consumed in the enzyme-processes. However, they may be retained, by modeling them with a time-delay-var with its specific formula. Note that the connected bioReservoir may be configured for the quantity of the bioPool to decay dynamically, representing their degradation and diffusion components, and it also may be consumed in other bioProcesses;

Effective-binding-sites (1418) is a simple float attribute of the subclasses of bioReactant that refers to molecules or complexes that have binding activity, such as all the subclasses of amplifier-bioreactant and leading-bioreactant, and its value is set by the modeler to indicate the number of effective binding sites per unit.

Kinetic-parameters are by default simple float attributes specific for some subclasses of bioReactants, which values are optionally set by the modeler to indicate the value of the kinetic parameter characteristic for that particular instance. A set of two attributes is provided for each of such subclasses, one to hold the absolute value and the other to hold a scaled value of such kinetic parameter, which are used by the set of absolute generic simulation formulas in conjunction with the absolute-valued variables, or by the currently default set of scaled generic simulation formulas in conjunction with the scaled-valued variables, respectively. Alternatively, in cases where the value of the kinetic-parameter is time-variant and dependent on other variable values, the modeler may choose for individual bioReactants to define for that attribute an instance of kinetic-parameter-var, in which case the modeler has to define the specific simulation formula to provide the value for that particular instance, as previously described. Examples:

- **Equilibrium-dissociation-constant** (1423) and **Scaled-equil.dissoc.k** (1424) for the subclasses of binding-bioreactant indicate the absolute or a scaled value, respectively, of the Ks characteristic of that agonist or complex-subunit;
- **Catalytic-constant** and **Scaled-catalytic.k** for the subclasses of enzyme.r indicate the absolute or a scaled value. respectively, of the characteristic kp of that enzyme, where kp is the equivalent of the units of activity per catalytic center;
- **Michaelis-constant** and **Scaled-michaelis.k** for the subclass substrate.r indicate theabsolute or a scaled value, respectively, of the Km characteristic of that substrate, where Km is the dynamic constant equivalent to the substrate concentration that yields half-maximal velocity. This attribute may optionally be considered to represent the Ks, where Ks represents the intrinsic dissociation-constant, when the system modeled is under rapid equilibrium conditions, which is usually not required because of the dynamic nature of the simulation system of this invention; or
- **Inhibition-constant** and **Scaled-inhibition.k** for subclasses of inhibitor-bioreactant indicate the absolute or a scaled value, respectively, of the Ki characteristic of that inhibitor or antagonist.

[0130] The **class bioProduct** has the subclass cell-bioProduct. As shown in FIG.15, a bioProduct (1518) is connected to an output p-connection at the bottom of a bioEngine (1501), and represents the material contribution from a bioEngine to the bioPool to which it is connected. To be operational, the bioProduct-post (711) connected to a bioProduct must have the same name as one of the bioPool-p-posts (712) connected at the top of a bioPool (713). Every instance of bioProduct (1518) has its menu (1519), and selecting "table" (1520) displays its attributes (1521), which in this case shows the default values for its attributes. Not previously described attributes specific of this class include:

- **Production-Rate** (1522) is given by a produc-rate-pvar (1523), which value indicates the rate of production of the units that will be added to the connected bioPool, which is dependent on the bioProduct's Stoichiometric-coeff and the Velocity of the bioEngine, and is an argument for the Input-rate of the connected bioPool.

• **Class bioRole-post**

[0131] A bioRole-post is located upon the subworkspace of a bioProcess and its function is to connect a bioRole-Object to a bioPool through a bioPool-post. The class bioRole-post is a subclass of bioPost, from which it inherits all its capabilities, and has two subclasses: bioReactant-post (703, 707), which together with a biopool-r-post (704, 708) distantly connects a bioReactant to a bioPool; and bioProduct-post (711), which together with a biopool-p-post (712) distantly connects a bioProduct to a bioPool. The new attribute defined for these classes is:

- **Ref-bioreservoir** (2234) is a pointer to the bioReservoir instance (2209) that encapsulates the biopool-r-post or biopool-p-post to which it is distantly connected. The value is interactively set by selecting the "set-refs" (2229) option, and it is used by the program for interactive navigation through bioReservoirs and bioProcesses.

BioModels

[0132] In this invention a bioModel means an iconic knowledge structure which, depending on the subclass, encapsulates in its subworkspace a set of interrelated composite bioObjects, such as bioReservoirs, bioProcesses or other bioModels. BioModels are used to partitioned the Virtual Model into a modular hierarchy of subworkspaces of bioModels, and can be also described as subsystems or fragments of a larger network of pathways, which can be reused as modules of desired degrees of complexity to be combined in a variety of ways to build larger and diverse systems. BioModels are objects represented by icons, and the constants, parameters, and variables that quantitatively model the system are distributed throughout their component building blocks. The dynamic aspect of a bioModel is determined by a system of algebraic and differential equations that provide the values for the encapsulated dependent and state variables. BioModels represent empirical biological models derived from experimental descriptive information and semi-quantitative or quantitative data. Very complex bioModels can be build by connecting component parts of an unlimited number of other bioModels by means of connection-posts. The icon of some bioModels, such as sequential time-compartments may contain stubs to allow connection to other bioModels.

[0133] The **class bioModel** is a subclass of bioView-Object and have no additional defined specific attributes, but inherit the attributes Names, Label, Description, References, Warnings, and Toggle-state. It comprises various subclasses which represent any physiological system at different levels of structural complexity, such as organ, tissue, cell-interaction (2501), cell (2602), or subcellular organelle (2406, 2416 - 2420), or any other intermediate compartment within those compartments, such as: submodel (2408), bioReservoir-Bin (2412), and On-Hold-Bin. A submodel schematically represents a portion of any size of the network of pathways of a larger structure such as a cell-bioModel. The subworkspace (2409) of a submodel contains a set of related bioProcesses and also a bioReservoir-Bin (2412), which encapsulates a set of related bioReservoirs connected to those bioProcesses. However, the bioReactants and bioProducts of any bioProcess that is a component of a bioModel may be connected to bioPosts of bioReservoirs that may be located in the same or in other bioModels. As the submodel becomes larger, it may be broken down into smaller submodels by simply cloning a new submodel and transferring to its subworkspace the desired bioProcesses and to the subworkspace of its bioReservoir-bin the corresponding bioReservoirs. The connectivity of the pathways is transparent to movements of bioObjects from one bioModel to another, since they are directly connected among them through the structures upon the subworkspaces of bioReservoirs and bioProcesses. This architecture results in a hierarchy of bioModels encapsulated within other bioModels, which ultimately encapsulate bioProcesses (2410) and bioReservoirs (2414), which encapsulate bioReactants, bioEngines and bioProducts (2411), or bioPools (2415), respectively. Although this topology may appear complex, at run-time each of the variables depend on the variables of only those bioObjects directly connected, and therefore the system operates as a set of processors concurrently computing in parallel. The distant connections between bioProcesses and bioReservoirs, established through the bioPosts anywhere in the knowledge-base, interconnect the different bioModels into a dynamic multidimensional network of any unlimited complexity. This architecture facilitates the implementation of feedback and forward loops, interactions between components of different pathways, elements that are shared by different pathways, and connections between segments of pathways that are built as separate reusable modules. The **class On-Hold-Bin** (2003) is characterized by having an activatable-subworkspace, which is usually maintained deactivated, and is used to store within a submodel bioReservoirs and/or bioProcesses that are either substitutions for the bioReservoirs and/or bioProcesses encapsulated in such submodel, or additional bioReservoirs and/or bioProcesses to extend the pathway represented by such submodel, which are appropriately connected, but which the user wants to exclude from a simulation or other uses. Some of those subclasses of bioModels, such as the sequential time-compartments, can be independently activated and activated by the program during a simulation run. The efficiency of large-scale simulation applications can also be significantly improved using that mechanism, by having whole branches of the model that are not relevant at some point in time deactivated and activated when required, driven by events generated as a result of the simulation itself, or at given time intervals.

[0134] The **class cell-bioModel** is used to represent a cell as observed from the inside (FIG.24). It is a container of subcellular compartments that may encapsulate different layers of submodels until at the end of the compartment hierarchy, the bioReservoirs and bioProcesses are encapsulated. Cell-bioModels are used for modeling prototypic cells of interest in a very detailed way through their functional structural components, which at the end encapsulate variables and parameters that allow to quantitatively simulate those models, or more realistically, constrained submodels within the cell by selecting any of the compartments at any desired level. Each cell-bioModel or any of its encapsulated bioModels can be used as modules to build other bioModels with increasing degree of complexity.

[0135] As shown in FIG.24, each cell-bioModel (2401) has a subworkspace (2402) upon which are represented the characteristic phases of the cell cycle (2421), that follow a repeated cyclic pathway, or the cell differentiate into a different stage (2428), or may go into apoptosis (2430), a terminal stage. The cell-phases (2404, 2421, 2428, 2430) are subclasses of the **class cell-phase**, the main defined subclass of the class **time-compartment**, but other types of compartmentalization are also possible by means of instances of the class **time-submodel** (2423, 2425, 2426). Instances of the **class biomodel-post** (2403, 2429), ant its associated methods allow to connect those cell-phases with another cells-phases upon other cell-bioModel, and to display them. The **G_0-compartment:** (2404) represents the background processes of resting-cells in addition to all those background bioProcesses that are not specific for any other phase of the cell-cycle or which crossover the boundaries of several of those phases. The **G_1-compartment** (2421) represents a state of cells after they have been activated by external factors, and may be further compartmentalized into two or more time-compartments: the **$G_{1.1}$-compartment** represents activated-cells or early-G_1-phase-cells as characterized by transcription of early-response genes, expression of new receptors; while the **$G_{1.2}$-compartment** or late-G1-phase-cells, is characterized by transcription of late-response genes or increased secretion of cytokines. The **S-compartment** represents the state of cells in S-phase, characterized by DNA synthesis. The **G_2-compartment** represent cells in the G2-phase, characterized by double DNA content and before entering mitosis. The **M-compartment** represents cells in the M-phase, during the mitotic process. The **Di-compartment** (2428) represents cells entering a new differentiation stage, when a different set of activated early-response and/ or late response genes induce permanent changes in the types of genes expressed. This layer is followed, depending on the combination and strength of signals provided by simulation events, by either the G0-compartment or both this and the G1- compartment of the next cell type in the differentiation pathway. The **Ap-compartment** (2430) for apoptotic cells, when cells in the G2-layer do not receive the appropriate signals to enter the M-layer. This is a terminal layer which only effect is to reduce the size of the total cell population. (Note that the total cell population size is increased at the transition after every M-layer, by an amount equal to the size of the population in that M-layer. The non-specifically-modeled overall cell death is represented by a decay parameter, set to fit the experimental data.

[0136] The default time-duration (317) of each of these temporal layers is set according to experimentally obtained knowledge of such duration, under specified "normal" conditions. A layer-rate-constant is constrained to values -1 > τ_1 > 1 and initially set to 0. When the effects of different combinations and strength of signals on the duration of any of the layers are measured, then the value of τ_1 is adjusted to fit the target values of the experimental set.

[0137] Each time-compartment (2404, 2423) encapsulates (2405, 2424) several spatial compartments that run in parallel to each other, and which represent the clearly defined subcellular compartments, called cell-Compartments, such as the cell-membrane (2406), cytoplasm (2416), nucleus (2417), endoplasmic-reticulum (2418), Golgi apparatus (2419), endosomes (2420), mitochondria, and so on. It also encapsulates a **timer** (2427) with an activation time attribute given by an elapsed-simul-time-par, which can be used in the control of the deactivation of the subworkspace that contains it by inference methods based Ion such value. The connections between the time compartments belong to the **class cycle-path** and encapsulate a number of parameters and variables to hold the values of the rate-constant and the progression rate, that can be used when developing quantitative models for simulation purposes. A number of other variables and parameters that describe among others: a) the size related attributes of the cells. and b) quantities related to population dynamics, such as those shown in FIG.3 (317 - 320), which are attributes of the cell-phases, are defined. A collection of inference blocks (Table 8) may also be used to control (Table 9) the activation and deactivation of the time compartments, at intervals that are predefined or dynamically computed depending on simulated variables are also defined. Table 7 lists as an example a set of simulation formulas for state variables that can be used to compute the dynamic changes in cell numbers that are accumulated in each of the cell-phases, when using these graphic structures in combination with a population dynamics approach. Furthermore Table 10 lists a set of rules that can be alternatively used to control the progression through those cell-phases, when using the mechanistic approach. As the reader may realize, there are a number of different alternatives, encapsulated within those graphic building blocks, to model the complex systems at different levels. Each of the values of the Cell-phase attribute of a cell-bioModel, indicates which of those phases is currently activated, if any in addition to the G0-phase. As shown in FIGs. 4, 28, and 29, there are switching points (406, 2803, and 2911) at early G1 where the cell has to decide between the differentiation and the cyclic pathways resulting in a variety of different patterns that depend on both the internal stage of the cells and other components in the system, external to the cells.

[0138] As shown in FIG.25, instances of the **class cell-interaction** (2501) encapsulate in their subworkspace (2502) the interaction of two cells (2503, 2504), that use the graphic extracellular-components to interact with each other directly by means of surface-components (2505, 2506); through the components that one secretes, the secreted-components (2507), for which the other has specific receptors, the surface-components (2508); and both interact with their environment through specific receptors (2509) for systemic-components (2510). As shown in FIG.26, each of those components have a table of attributes (2616, 2620, and 2633), referring to an specific bioEntity (2624), bioReservoir (2626) and bioProcess (2628) within the cell-bioModel, and are connected to the internal mechanistic pathways. Clicking on those icons displays their menus (2614, 2618, 2631) from which options the user can select the display of the associated structures, as specified in the attributes. The different menu options and their associated procedures allow the modeler to automate the task of establishing the distant connections to those graphic structures. The interacting-cells (2503 and 2504) in this interface do not encapsulate a mechanistic model of its components, but rather refer to and display another cell-bioModels that store those mechanistic models. In this way, this structure is used as an interface to design interactions between other cell-bioModels, and new cells and other connections to bioProcesses or bioReservoirs can be establish. The external components (2612), such as serum or incubation media, or single or lumped toxic-byproducts, for which there are specified or lumped receptors are represented by a set of bioReservoirs, such as serum-Pool, are connected to a bioProcess with a receptor-engine that has at least one bioProduct connected with one of the bioReservoirs within the cell. The function of the bioEngine is to compute the consumption an exhaustion of nutrients at a rate proportional to μ, the specific growth rate. The byproduct-Pool's concentration is used to modify the value of μ by multiplying it by a parameter $0<m<1$.

[0139] Cells are represented in this invention in two different ways: as the class cell-bioModel (2401) composed of cell-Compartments, and as the **class cell** (2602) used to represent a cell as observed from the outside and including any defined, mostly external, characteristic markers (2607), as shown in FIG. 26, that can be used to assigning that cell to one type of cells and to distinguish it from cells of other types. The instances of class cell (2602), with sets of stubs on the top and bottom of their icons, are used to classify cells according to a branching differentiation tree, as shown in FIG. 27. to define the many nuances between them that differentiate them from each other, particularly when following the almost continuous differentiation process. As an example, a Th2-lymphocyte(2607) is shown in FIG.26. An instance of cell may correspond to an instance of cell-bioModel, but it is expected that there are many more intermediary stages represented by the lighter cells, which represent the much lower level of detail most frequently available. BioReservoirs representing cell populations may refer in their Ref-bioentity attribute to a named instance of cell. One of the menu options for this class is "biomodel" which upon selection displays the subworkspace of the bioModel referred to by one of its attributes, **Ref-biomodel**, if any, and that option is only visible when such attribute has a value. The **class differentiation-pathway** provides structures that allow the modeler to manually build differentiation trees, which are based on expert-knowledge to represent the sequential progress from one stage of differentiation to the next. The different stages through which a cell lineage goes through is a continuous process, which can be simplified with the representation used in this system consisting of discrete stages that represent different stages of differentiation, defined by characteristic and measurable phenotype, that follow a tree like longitudinal pathways. The cells in those pathways are of the class cell (2602) further encapsulate detail as shown in FIG.26. A cell contains information in a table of attributes (2605) and in its subworkspace where the different icons include a cell-surface (2608), since this is a representation of the cells from the outside, and a set of components that allow that cell to display characteristic markers and receptors that allow them to interact to the outside world.

Inference and Simulation Structures

[0140] **Functions** are statements which define a sequence of operations that are performed when the function's name and arguments appear as part of an active expression. The tabular-function-of-one-argument, a class defined within the Shell, has an important use in the domain of this invention by allowing to deal with situations when the algebraic relationship between two variables is not known, but the experimental data is available, and straight-line interpolation is optional. Tabular functions can also be used in other expressions, such as the rule: if the [Productl]([Substrate1]) decreases then inform the operator that "[Substrate-1] may be inhibitory at high concentrations".

[0141] **Relations** are defined associations between two items which can be dynamically concluded and reasoned about, and in the current implementation of this invention, they represent physical and abstract relationships between objects, such as a-downstream-bioReservoir-of, or the-downstream-BR-list-of. After being defined by the developer, relations are established transiently at run-time by the inference engine. The dynamic reasoning used for query, navigation, and simulation depend heavily in the use of relations, in addition to the graphical connections. The existing relations for a particular object can be also be listed using the "describe" option. Most of the relations are created during the initialization set of procedures, while others are created at run-time.

[0142] Domain-specific **rules** are used to conclude or to respond to the changing conditions in the world defined by the domain-specific knowledge structures. Such events may be a relation being established or broken, or moving the icon by the user which is frequently used in this application to either trigger the automatic configuration of the value of some attribute, or the appearance of the icons, or the position of the icons.

[0143] **Formulas** (Tables 1 through 7) are expressions that define the relationships of a variable or parameter to other variables or parameters, in a way that when one variable or parameter changes the formula predetermines in which way other variables or parameters will be changed, either by the inference engine, in the case of general formulas, or by the simulator, in the case of simulation formulas. Generic simulation and general formulas provide values for a whole class or group of bioVariables or bioParameters, in contrast with the specific simulation and general formulas, which are attached to one variable and apply only to that variable. The generic simulation formula for sets of scaled-valued variables (Tables 1 and 2), mixed-type variables and absolute-valued variables (Table 5) will be discussed under the Simulation Mode heading.

2. Modeler Mode: Creating the Virtual Models Menus, Palettes and Libraries

[0144] The Modeler Mode is made available to those users who have rights to build or modify applications. The modeler's tasks are all based on the basic paradigm of "Clone, Connect, Configure and Initialize", and consist of building and configuring graphical models of the structure of bioEntities, as well as to design and build graphical models of complex networks of cross-talking pathways, which result from the modeler's actions of cloning, connecting and configuring sets of bioProcesses, bioReservoirs, and bioEntities. To facilitate the modeler's tasks, the Modeler-Menu provides a way to organize and access the different components of the system.

[0145] FIG.16 shows the **domain-menus** associated with this mode, shown here pulled-down for demonstration purposes. Selecting the "Help & Mode Menus" option (1601) allows access to the available user modes, which upon selection causes a change to that mode of operation and displays the specific domain-menus for the window from which it is selected. Selecting Modeler Mode (1602) displays the top layer of menu heads which additionally comprise the options "Palettes" (1603), "Structure Libraries" (1619), and "Pathway Libraries" (1628), allowing to access parts of the Virtual Models which usually requires navigating through a hierarchy of walking menus. The discussion of this menu system serves the purpose of discussing how the building-blocks are organized in Palettes and how the iconic models build by the modeler are organized and stored in Libraries. Palettes contain domain-specific and generic building-blocks built by the developer and to be used by the modeler. while Libraries contain application-specific models built by the modeler and to be used by different types of users. A facility is provided, the Modeler-Palettes-Bin, to automatically create new empty Palettes to be populated by modelers, to allow them to expand and complement the built-in library of building-blocks.

[0146] The **Palettes** provide a large number of characteristic types of bioObjects, grouped by subclasses and organized in special subworkspaces of buttons organized in the subworkspace of a "Palettes-Bin". Selection of any bioObject upon a Palette in Modeler Mode implies clone, which automatically results in a clone of such bioObject (including its subworkspace and all structures upon it, and all the further levels of encapsulation) to be attached to the mouse-pointer and ready for the modeler to transfer it to the desired subworkspace. Selecting "Palettes" (1603) pulls down its options, which may point directly to the palettes of important classes of bioObjects and model-blocks, such as 1604 > 2001 and 1608 > 1901. In the case of classes with large number of building-blocks, those may be grouped further in a number of palettes that can be accessed through walking menus, such as when selecting "Molecules" (1609) displays a pull-down menu headed by "Molecules" (1610), which options may now point directly to Palettes.

[0147] **Libraries** comprise among others two types:

Structure Libraries, are hierarchically organized workspaces (1623) containing models of specific complex-bioEntities (1624) that model the functional structure of molecules or molecular-complexes at various levels of complexity, and which can be accessed by selecting the options of the menu head "Structure Libraries" (1618). Selecting "Cytokine Library" (1620) displays the menu (1621) which options refer to libraries of different families of cytokines, and further selecting one such as "IL4.like Library" (1622) displays the library (1623) of the IL4 family, which contain several cytokines with similar structures, which prototype is IL4 (1624), which subworkspace contains several domains, mofifs and groups. Those icons may be used to: a) be cloned and modify to build different but related structures; b) be placed in the subworkspace of protein-complexes, as shown in FIG.18; or c) they may be preferentially named and copied, and the dummy copies that refer to the original are used for several of the more complex structures.

[0148] **Pathway Libraries**, are hierarchically organized workspaces (1630) containing sets of related bioProcesses, such as those contained in the subworkspace of a submodel, and a bioReservoir-Bin with the bioReservoirs connected to those bioProcesses, where those bioProcesses and bioReservoirs are connected forming the nodes of a pathway that may be or not multidimensionally branched. The pathways included in the libraries are those that are common to several systems, although they may require some modifications, and they can be cloned instead of being built from scratch. Alternatively, the "Pathway Libraries" (1628) menu may be used to hierarchically access any submodel of the Virtual Model, as an alternative to interactively navigate through all the layers of subworkspaces involved. The options of the menu head "Pathway Libraries" (1628) point directly to submodels which subworkspaces are displayed. For example, selecting "EGF.R.Pathway" (1629) displays the components of the initial specific steps of such pathway (1630) containing bioProcesses (1631) and a bioReservoir-Bin (1632). Modelers can also clone submodels, in which all the bioProcesses organized in its workspace, together with the contained bioReservoir-Bin and all the bioReservoir organized in its workspace, are also cloned preserving the labels and any other attributes, but with no names, since names are unique. All new bioReservoirs and bioProcesses have to be named (with the same basic name and different tags), and the connections have to be reestablished by naming the bioPosts and setting the ref- attributes.

Creating New Composite Objects

[0149] There are several alternatives for building new generic types of composite bioObject. One method comprises the following steps: a) an instance of the desired class, such as bioProcess, is created by selecting from the menu of the object-definition of that class the "create-instance" option and transferred to a workspace; b) the subworkspace of such instance is created by selecting from its menu the "create-subworkspace" option; c) the desired set of instances of characteristic classes of visual Objects are created or cloned and transferred to such subworkspace; and d) the newly created instances are positioned and connected in specific ways, to conform to the designs specified in this invention. A preferred alternative way is to "clone" an existing composite bioObject that is similar to the desired one, and delete and/or add components, as desired. The additional generic building-blocks are added to palettes accessible through the domain-menus, and are used to create new instances of the same type by cloning such generic structures. The modeler can then further individualized each newly cloned instance by configuring the attribute tables of the superior object, or by configuring the attribute tables of the objects encapsulated within that superior object, either as attribute objects or as objects upon successive subworkspaces.

[0150] Furthermore, it is possible to create transient objects at run time in two ways: a) by using the "create a <class-name>" and "transfer" actions; or b) by using the "create a <class-name> by cloning <instance-name>" and "transfer" actions, which is the equivalent to the clone option described above, and is the preferred way for composite objects.

Transient objects can be made permanent by using in addition the "make <instance> permanent" action after the instance has been transferred to a workspace. This invention makes extensive use of dynamically creating objects at runtime by cloning, such as when creating dynamic pathways, dynamic query-panels, dynamic query-output-panels, dynamic entry-panels, dynamic graphs, dynamic lists, and so on. A set of partially prebuilt complex sets of structures, such as those required for the different types of panels, are organized and stored in the subworkspaces of objects called bins, and used as master structures to be cloned and completed with additional context-dependent structures at runtime.

Modeling bioEntities

[0151] The modular functional representations (FIG.18) of structure that are novel teaching of this invention allow to represent the modular structural patterns frequently used by nature. The iconic structural representation is used to abstract the detailed structure of different molecules to highlight their similarities, and to show that even prototypes for different families present several similarities. Abstraction is obtained in part by selectively hiding some detail in the subworkspaces of other components focusing on components that relate to their function, to facilitate reasoning about structure-function relationships. The components of a molecule are used by the user to reason about its function, since the presence of given domains, motifs, or simple-bioEntities usually is associated with described function, therefore directing the scientist's attention to further focus on those functions. Each of those bioEntities have their associated tables of attributes, where the modeler can include additional information, data, and references to access other sources of information about those models, which can be integrated with the function of those bioEntities provided by the pathways related models. That knowledge integration is performed sometimes interactively by the user and sometimes programmatically as a result of a request by the user. Each superior bioEntity represents the structure of a biochemical physical entity in a particular state which corresponds to the concept that different bioReservoirs hold molecules or complexes in different states. The structural information as provided in the corresponding bioEntities is supplemental and is not necessary for creation or navigation thought the pathways, or for simulation. This information is used independently of the pathways, or they may be used in combination, as in the complex queries.

[0152] The **Palettes** with representative examples of master instances of some of the different subclasses of protein-domain, protein-motif, and simple-bioEntity, are accessed through the walking menu of "Protein Components" (1616). Some of those building-blocks are very generic or "empty" structures while others may contain more or less detailed prototypic structures, in which case they are named master-structures that may be referred to by the master-bioentity attribute of its copies. Each instance of these components have an associated menu (1805) from which options specific for each class can be selected to perform generic or specific tasks. Selecting the "table" option (1823) shows their tables of attributes providing options for the modeler to include additional information specific for each instance or copies. Clones of those instances can be combined in different ways to model, using the basic paradigm of "Clone, Connect, Configure and Initialize", at the desired level of detail, the structure of a bioEntity.

[0153] In addition to the standard Shell's "clone" capability, there is a novel domain-dependent capability associated with the class complex-bioEntity and all its subclasses. Selecting the "create-local" option (1824), restricted to Modeler Mode and when the bioEntity has a name, starts the et-create-local-proc and allows the modeler to create local copies from any instance. The term clone means in this invention a complete reproduction of the original, with exception of the name, while the term copy means an incomplete reproduction that is usually stripped from the contents of the subworkspace of the original, but has a pointer to the original and is used in conjunction with it. The master-bioentity attribute of the copies is automatically set to refer to the named master-structure. The dummy copies refer to the original when the "master-details" is requested, which subworkspace with the prototypic structure is displayed. Local copies of any complex bioEntity may have private specific information in its table of attributes and in its subworkspace, shown by selecting the "local-details" option. Simple-bioEntities do not have a subworkspace and do not have master-bioentity attribute. This implementation is preferred to avoid redundancy and allowing modification at only one site when the details of the structure have to be modified.

[0154] As shown in FIG.18, the subworkspace (1807) shown for a multi-unit receptor-complex (1804) contains the icons of its various components, which may be copies, such as those referring to structures in the Libraries (1808, 1815, and 1817), or may be originals, such as for the simple components or some specific complex components (613, 619 and 621 are originals within the original master structures). Upon clicking on each of those components, their subworkspaces or the subworkspaces of their master structures are displayed on the screen, showing their structural composition. The level of detail is optional, and components at different levels in the hierarchy of subclasses can be connected to each other. For example, clicking upon a subunit of the receptor (1808, a copy) shows its structure (1809, from the master) composed of domains, other simple-bioEntities, and other optional auxiliary structures, such as those that may encapsulate: notes about its function (1810), sequence (1811), references (1812), or a GIF image of its three-dimensional structure. Any of those components may encapsulate more detailed information, such as that displayed when clicking in one of those domains (1813), which structure (1814) is composed of other domains, spacers and other simple-bioEntities. Back to the main structure, clicking on a kinase (1815, copy) that form part of the complex, its structure (1816, from the master) may be represented by its two subunits, a regulatory-subunit and a catalytic-subunit. Or the structure can be directly represented by it lower level components, such as when clicking on other associated kinase (1817), which structure (1818) is composed directly by several domains and other simple-bioEntities. Any of those molecular components (1819)may encapsulate any type of other relevant information (1820) which allows to represent all members of the family by just one structure, by describing additionally the differences between the different members.

Modeling Pathways

[0155] The **bioReservoirs Palette** contains prebuilt default instances of: **bioReservoir-Bin**, used to store a set of related bioReservoirs, and a set of different **prototypes of bioReservoirs**, representing all the subclasses which encapsulate one corresponding bioPool, and in the color of the type-color region of their icon. Each of those prototypic bioReservoirs comes with a minimal set of basic structures, and model-boxes and additional bioPosts are added interactively when needed. By cloning a bioReservoir from its Palette and transferring it to the subworkspace of a bioReservoir-Bin, it can then be connected to a bioProcess. Another alternative for adding bioReservoirs to the Virtual Model is to clone existing bioReservoirs with similar characteristics, by selecting the "clean-clone" option (1239) from their menu, and configure the new name and desired attributes. The "clean-clone" option starts the BR-clean-clone-proc, which creates a clone of such bioReservoir and sets the values of all the attributes of all structures involved to their default values. Various menus options and associated methods are offered to automate the task of establishing those connections.

[0156] The modeler can also model inputs or outputs to a bioPool as mathematical models using model-blocks (1222) held in **model-boxes** (1218). A set of menu options, restricted to Modeler Mode, are defined for the class bioPool to automate the task of adding a scaled-input-model-box (1218), input-model (1236), or output-model-box (1237) when desired. The options labeled "add-scaled-input" (1325), "add-input" (1326), and "add-output" (1327), start their associated methods, create-scaled-input-model-box-proc, create-input-model-box-proc, and create-output-model-box-proc, respectively, which result in the cloning of the corresponding prebuilt master model-boxes, and then transferring the new instance to its specified position in relation to the bioPool. The modeler has only to extend its stub to join it with the color-matched stub of the bioPool. Each of those named master structures is simply a generic model-box, each of the subclass referred to by its name with a color-coded subworkspace containing a connection-post which attribute Superior-connection has been set to refer to the only connection stub of the superior model-box. Any model-block connected to such connection-post of a model-box is connected also with any object to which the model-box is connected, resulting in the ability to establish a connection between an encapsulated model-block and a bioPool, by cloning the desired model-block from the Palette and connecting to the connection-post of a model-box connected to that bioPool. The generic formulas that provide the value for the Accumulation of each bioPool already integrates the output of any such connected model-block as either an absolute input, a scaled input or an output, depending on the class of the model-box that encapsulates the model-block.

[0157] As shown in FIG.19, the **Model-blocks Palette** (1901) is provided to the modeler with a default instance of each of the subclasses of model-block. Some examples of the attributes for some subclasses with their default values are shown through the tables of attributes of: constant.f (1902), exp.c.pdf (1903), exp.rn.pdf (1904), unif.c.pdf (1905), sigmoid.f (1906), and generic.model.block.f (1907). The latter is a very generic block that allows the modeler to create instances and configure as many attributes as required by defining their slots with any desired type of value, including instances of model-block-var, in which case their specific formula would also have to be defined. The output-1 of this block is also a variable for which the modeler has to write its formula, to incorporate as arguments the newly configured attributes

[0158] As shown in FIG.20, the **bioProcesses Palette** (2001) provides the most representative types of generic bioProcesses, prebuilt and ready to be cloned and further configured. These preassembled and stored structures allow the repeated use of bioProcesses as building blocks of more complex pathways. Like in chemical systems, some bioProcesses are relatively simple and some may become more complex, but all are built with the basic building blocks (2504 -.2511). The modeler can, for example, clone a R2.L1, which subworkspace (2008) comprises one receptor-bioengine connected with two receptor.r and one ligand.r at the top, and one bioProduct at the bottom. After transferring the clone to the desired workspace (2409), the bioProcess can now be configured, by clicking on its icon and selecting its "table" (1403) of attributes. Selecting "details" (1416) provides access to all its components, which can now be configured by accessing their tables attributes. An alternative to the Palette is to clone another existing bioProcess with similar characteristics and labels directly from a submodel, and then transferring them to the desired submodel and the configuring its components. Two menu options are provided to facilitate that task, "labeled-clone" (1427) and "clean-clone" (1428), which start either BP-labeled-clone-proc or BP-clean-clone-proc, respectively. The first procedure creates a clone of the bioProcess, and then sets the values of all the attributes except the label of all structures involved to their default values. The second procedure calls the first one, and also sets the values of the labels attribute of all structures involved to their default values.

[0159] The **bioEngines Palette** and the **bioReactants Palette** are provided for the purpose of allowing the modeler to: a) modify individual instances of bioProcesses cloned from the Palette, to accommodate particular infrequent needs: and b) to design new prototypic bioProcesses. by starting from a clone of an existing bioProcess and adding or removing bioReactants or bioProducts, or by exchanging the bioEngine for another of compatible type. The modeler may want, to modify the Velocity attribute of any bioEngine. and to provide specific formulas to represent their favorite equations to best describe the process being modeled, which have to refer to the specific set of bioReactants connected to that bioEngine. Such newly designed bioProcesses may be stored as building-blocks in the appropriate Palette within the Modeler Palette Bin, as described. In addition, the modeler may use a tabular-function as a look-up-table, from which the program is able to interpolate, to model a process by entering experimental pairs of values of the independent and the dependent variables in such table, such as a dose response. To facilitate that task, the option "create-sw" (1526) is defined for the class bioEngine that starts the create-sw-for-bioengine-proc, which creates a subworkspace for the bioEngine by cloning the subworkspace of the bioengine-sw-master, which contains a tabular-function-of-1-argument and instructions for the modeler about how to use such a tabular-function.

[0160] As shown in FIG.22, several steps are required **to establish the connection** between a bioReactant or a bioProduct and a bioPool. To facilitate this process and to reduce the sources of error while entering those values by the modeler, various methods are offered to automate the tasks of: creating and naming, or removing the names, of bioPool-posts, removing the names and the references of of bioRole-posts. and to set the appropriate values for the ref-bioprocess of bioPool-posts and ref-bioreservoir of bioRole-posts, as following:

The first step is to select the bioProcess (2201) and bioReservoirs (2205, 2207, and 2209) to be connected, if already in existence, otherwise they are created by cloning from the Palettes and named. BioProcesses and bioReservoirs have to be named before their bioPosts can be named. By selecting the "details" option (2203) from the menu (2202) of the bioProcess, and such of the bioReservoirs, their subworkspaces (2204, 2206, 2208, and 2210, respectively) are displayed. Selecting the menu (2214) option "add-top-post" (2215) of the bioPool (2213) starts the create-biopool-p-post-proc, which first scans for the first free stub of those defined at an even-numbered port (p02 to p12) and a) if one is found, it creates an instance of biopool-p-post (2217) and aligns it opposite to such stub within the first layer. b) if none is found, it then scans for the first free stub of those defined at an uneven-numbered port (p01 to p11) and, if one is found, it creates an instance of biopool-p-post and aligns it opposite to such stub within a second layer (more distant from the bioPool) of such bioPool-posts, and if none is found, then: i) it moves the existing biopool-p-post connected at the first port (p01) to a more distant position, to start the next layer, ii) it creates an instance of biopool-p-post and transfers it to the next free predefined position within that layer: and c) it sends a message to the modeler with instructions for making the specific connection. Then it calls the pool-post-name-it-proc and returns. The "add-bottom-post" (2216) functions in a similar way, but it adds instead biopool-r-posts at the bottom of the bioPool.

[0161] To connect each of the bioRole-posts (2212, 2218, and 2225) to a complementary and not yet connected bioPool-post of the appropriate bioReservoir requires to create a new bioPool-post of the appropriate subclass and to establish the connection to the bioPool by extending its stub and joining it to the closest free stub on the bioPool. Here is shown how a distant connection is established between and already named and connected bioPool-p-post (2219) and the bioProduct-post (2225). Clicking on the bioPool-p-post (2219) in Modeler Mode displays its menu (2220), and selecting "table" (2221) shows its attributes (2222). The tables for a bioRole-post (2222) and a bioPool-post (2228) are shown with all the values already set. At this point in the connection process, the values for the Ref-bioprocess (2235) would be the default value none, while the values of the Names (2223) and the label (2224) attributes will be already set, with the value of the Label set to be the same as its name in the case of all bioPosts to be used at run-time when using the alternative simulations procedures. To give the bioRole-post (2225) the same name as the name of the bioPool-post (2219) to which is to be connected is accomplished by selecting either the "name" option (2227) or the Names attribute value slot (2232) of the bioRole-post (2225) and clicking on the name-display (2228) of the bioPool-post (2219), since name-display has text-insertion capabilities and automatically inserts its text. This completes the basic connection process that allows the inference engine and the simulator to reason about those connections. Once the bioRole-post (2225) and the bioPool-post (2219) are equally named, the Ref-bioreservoir attribute (2234) of the bioRole-post and the Ref-bioprocess attribute (2235) of the bioPool-post are automatically set to refer to each other's superior bioObjects by selecting the "set refs" option (2229) of the bioRole-post, which is only available when it has a name. These attributes are used by the program when the user selects the "show-br" option (2241) or the "show-bp" option (2242), which are available in the menus (2220 and 2229) of bioRole-posts and bioPool-posts, respectively, only when those attributes have a value.

[0162] Connections can be changed and previously connected bioPosts can be recycled by using the options "un-name-it" (2219) of the bioPool-post which only appears if it has a name, "unname-it" (2229) of the bioRole-post which only appears if it has a name, and "name-it" (2223) of the bioPool-post which only appears if it has no name, followed by clicking the name of the bioPool-post into the name slot of the bioRole-post and selecting the "set-refs" (2228) of the bioRole-post. The option. "name-it" (2240) starts the pool-post-name-it-proc which gives the Names of the bioPool-post a name composed of the base name of the superior bioReservoir followed by the default value of the Label (2224), which is a reference to the class and relative position of that bioPool-post. The option "unname-it" (2236) associated with the class bioPool-post starts the poolpost-remove-name-proc which changes the values of the Names and the Ref-bioprocess of the bioPool-post to their default value none, removes the base-name from the value of the Label (2224) restoring it to its default value. The option "unname-it" (2237) associated with the class bioRole-post starts the rolepost-remove-name-proc, which changes the values of the Names (2232), Label (2233), and Ref-bioreservoir (2234) attributes of the bioRole-post, and the Ref-bioprocess (2235) attribute of the bioPool-post, to none.

Biochemical Modeling Techniques

[0163] To simplify the overall structure of the system, or when the intermediary detail is unknown, it is possible to consider parts of such pathways as **black-box-bioProcesses** with inputs and outputs connected to other modeled pathways or to other black-box-bioProcesses, which are not modeled in detail. If and when it is desired to model disturbances at specific points in the pathway represented by a black-box-bioProcess, it can be split into the pathway before and the pathway after the regulated location, which is represented by a normal bioProcess, with the two new pathway segments now represented by two new black-box-bioProcesses, and the segment where the disturbance occurs to be defined and modeled. In addition, it may be desirable to model a pathway as a single **lumped bioProcess**. Such may be the case for an enzymatic pathway for which: a) the details may be unknown or not desired, b) only the inputs and the outputs are relevant, and c) sufficient quantitative or qualitative information is available about the rate limiting enzymes and the inputs. For these cases, the currently preferred embodiment provides enzyme-bioProcess subclasses with two or more enzyme-bioReactants, with their Velocity depending only the Concentrations of those enzymes, together with the substrates that are inputs to the pathway and the products are outputs of the pathway, ignoring the intermediaries and other participating enzymes.

[0164] For bioReservoirs that represent the input into the **system** of a synthetic agonist or antagonist, the Concentration has a normal default initial value of 0.0, it has no modeled inputs from bioProcesses, and it may be modeled by a model-block. However, quantities of that compound may be transferred to another regular bioReservoir in a different compartment. It may not be desired in most occasions that the synthesis of macromolecules be visually modeled by bioProcesses, because of its complexity and the large number of steps or components involved. In these situations, the synthesis may be modeled by adding a model-block to one input-box connected to the bioPool, which is represented by the additional term already integrated in the Accumulation equation. The output of this model-block may define a global synthesis-rate that may be made dependent on the values of any other variable of the system multiplied by a synthesis-rate-constant parameter, a separate attribute of the model-block which may also be dynamically increased or decreased as a result of the activation or deactivation of other bioProcesses.

[0165] When **degradation or diffusion** are important components of the model, they can be separately modeled by using additional bioProcesses to model those processes, adding a model-block to the output-box connected to the bioPool, or using defined degradation-rate or diffusion-rate parameters, respectively...Processes in which the **binding** step is the limiting factor (the catalytic or other actions are fast so that the ES complex is very short-lived), are usually modeled by a single bioProcess. If the complex stays around exposed to other interactions, two bioProcesses may be used: complex-formation and catalytic or other action, each with its respective constants. **Inhibitors and antagonists** should be modeled by separate bioProcesses. The intermediary complexes that are formed in enzymatic reactions, and their concentrations such as $[ES_1]$ or $[ES_1S_2]$, are preferably not explicitly represented, unless they are the specific targets of regulation or translocation to different compartments. Otherwise they are implicitly represented within the model equation of the bioEngine. Reaction systems with two or more substrates of the **ordered or pingpong** types, may be represented by either one bioProcess with two substrate.r, in conjunction with the appropriate Velocity equation, or by two separated bioProcess in series. In reactions with cooperativity, such as allosteric reactions of an enzyme with n equivalent substrate binding sites, can be either modeled by: a) one bioProcess with a 1E.nS.1I.HMME bioEngine with each of the n substrate.r representing the same bioPool, each with a stoichiometric coefficient equal to 1, but each with a different effective Km corresponding to each additionally bound molecule, or b) one bioProcess with a 1E.1S.1P.0I.HE bioEngine with one substrate.r with stoichiometric coefficient equal to n, and the Velocity formula representing a Hill equation. **Allosteric** reactions of an enzyme with binding sites for two or more different ligands, either substrates, inhibitors or modifiers, can be either modeled by: a) one bioProcess in which each ligand is represented in the formula for appropriate class of variable providing the Velocity, or b) two or more bioProcesses, in series or parallel or both, when the mechanism of the reaction is known and information is available.

[0166] Different types of **inhibitors** may be modeled in a variety of ways to show different behaviors, usually involving a complex-form-process in which the resulting complex would have different characteristics: A **competitive** inhibitor is best modeled as one enzymeEngine comprising enzyme.r, substrate.r and inhibitor.r with its matching standard kinetic equation. An **irreversible** inhibition is equivalent to enzyme removal from the system, with the Vmax reduced, and it is therefore best represented by two parallel bioProcesses: an enzymeProcess for $E + S \rightarrow P$, and b) a complex-form-process for $E + I \rightarrow EI$ It may also be represented by one enzyme-process encapsulateing enzyme.r, substrate.r and inhibitor.r, and two bioProducts ($E + S + I \rightarrow P + EI$), with its matching standard kinetic equation. An **uncompetitive** inhibitor may bind reversibly to the ES complex yielding inactive ESI complex. This type of reaction may be represented by three bioProcesses, first in series and then in parallel: a) complex-form-process: $E + S \rightarrow ES$; b) enzymeProcess: $ES \rightarrow E + P$; and c) complex-form-process: $ES + I \rightarrow ESI$. It may also be represented by one enzyme-process comprising enzyme.r, substrate.r and inhibitor.r, and one bioProduct ($E + S + I \rightarrow P$), with its matching standard kinetic equation. A **noncompetitive** inhibitor and the substrate do not affect each other's binding to the enzyme, however the trimeric complex SEI ($ES + I \rightarrow SEI$ or $EI + S \rightarrow SEI$) may be inactive. The main effect is a sequestering of either enzyme or both enzyme and substrate, and the Vmax is reduced. This type of reversible competition is best modeled by three different bioProcesses: an enzyme-process $E + S \rightarrow P$ (a), and two complex-form-process: $E + I \rightarrow EI$ (b), and $E + S + I \rightarrow SEI$ (c), where (a) and c) have the same Ks and (b) and (c) have the same Ki. It may also be represented by one enzyme-process comprising enzyme.r, substrate.r and inhibitor.r, and three bioProducts ($E + S + I \rightarrow P + EI + SEI$), with its matching standard kinetic equation. A **mixed-type** inhibitor I is such that reversible binding of either I or S to E changes the Ks or Ki of the other by a factor α, yielding ES, EI and inactive ESI complexes. It may be best represented by three different bioProcesses: an enzyme-process $E + S \rightarrow P$ (a), and two complex-form-process: $E + I \rightarrow EI$ (b), and $E + S + I \rightarrow SEI$ (c), where (a) and (c) have different Ks and (b) and (c) have different Ki. The Ks' of (c) may equal the Ks of (a) multiplied by α, or similarly, the Ki' of c = α*Ki, where a is a cooperativity-coefficient that may be introduced as a factor in the formulas of the respective Ks or Ki.

[0167] Different types of **feedback regulation** by downstream products is represented by different sets of bioProcesses. As examples are given the following cases of **feedback inhibition by more than one product**.

[0168] **Cooperative** or synergistic inhibition is when mixtures of P_1 and P_2 at low concentrations are more inhibitory than the sum of inhibitions by each at those concentrations, when both P_1 and P_2 can combine with E simultaneously to form EP_1P_2 complexes. This system may be represented by 4 bioProcesses: an enzyme-process: $E + S \rightarrow P$ (a), and three complex-form-processes: $E + I_1 \rightarrow EI_1$ (b), $E + I_2 \rightarrow EI_2$ (c) and $E + I_1 + I_2 \rightarrow EI_1I_2$ (d), where $I_1 = P_1$ and $I_2 = P_2$, and where (d) have different K_{i1} and K_{i2} than (b) and (c) (the K'_{i1} of (d) = α*K_{i1}, and the K'_{i2} of (d) = β*K_{i2}, where α and β are cooperativity-coefficients). **Cumulative** or partial inhibition is when each end product is a partial inhibitor, and a saturating level of one alone cannot drive the velocity to zero. Each of I_1 or I_2 alone, either increases the Ks of S (partial competitive inhibition) or decrease kd of E (partial non-competitive inhibition) or both (partial mixed-type inhibition). This type of reversible competition is best modeled by three different bioProcesses: an enzyme-process: $E^{-} + S \rightarrow P$ (a), and two complex-form-processes: $E + I_1 + I_2 \rightarrow EI_1I_2$ (b), and $E + S + I_1 + I_2 \rightarrow ESI_1I_2$ (c), where (a) and c) have the same Ks and (b) and (c) have the same K_{i1} and K_{i2}. It may also be represented by one enzyme-process comprising enzyme.r, substrate.r, inhibitor1.r and inhibitor2.r, and three bioProducts ($E + S + I_1 + I_2 \rightarrow P + EI_1I_2 + ESI_1I_2$), with its matching standard kinetic equation. **Concerted** or multivalent inhibition is when both end products have to be present to be inhibitory, and either alone has no effect on E, when either EI_1I_2 does not bind S or ESI_1I_2 is inactive. It may be represented in two alternative ways, either by: 1) two bioProcesses: one enzyme-process: $E + S \rightarrow P$ (a) and one complex-form-process $E + I_1 + I_2 \rightarrow EI_1I_2$ (b); or 2) one enzyme-process: $E + S + I_1 + I_2 \rightarrow ESI_1I_2$ (c). In both (b) and (c), the matching standard kinetic equations include a term to represent the concerted effect of I_1 and I_2, that is proportional to the Concentration of I_1 multiplied by the Concentration of I_2. **Additive** inhibition implies the presence of two distinct enzymes or catalytic sites, each sensitive to only one of the feedback inhibitors. This type of reversible competition is best modeled by three enzyme-processes: a) $E + S_1 + S_2 \rightarrow P$; b) $E + S_1 + S_2 + I_1 \rightarrow P$; and c) $E + S_1 + S_2 + I_2 \rightarrow P$. It may also be represented by one enzyme-process comprising enzyme.r, substrate1.r, substrate2.r, inhibitor1.r and inhibitor2.r, and one bioProduct ($E + S_1 + S_2 + I_1 + I_2 \rightarrow P + EI_1I_2 + ESI_1I_2$), with its matching standard kinetic equation. **Sequential** inhibition is when downstream of E_1 and S_1, one product P_1 inhibits enzyme E_2 and another product P_2 inhibits enzyme E_3 of another pathway of utilization of S_1. When both P_1 and P_2 are at high levels, S_1 accumulates, and S_1 may also inhibits E_1. This type of feedback inhibition is best represented visually by a combination of the appropriate bioProcesses, in parallel and in sequence as required.

Handling Challenging Aspects of Modeling Complex Systems' Behavior

• Modeling Internal Mechanisms of Cells

[0169] The transfer of molecules from compartment i to compartment j does not happen at once, but rather follows a Gaussian distribution, which may be modeled using the corresponding model-block from which the translocation-rate would be dependent. The analytic equations for the distribution of bioEntities in consecutive pools, are a function of the initial concentration of the first bioPool and the corresponding translocation-rate constant (k,1507) and time-delays (d, 1510), respectively, of the successive bioEngines. For example, to model the series of translocations of proteins through the several compartments in which they are processed after their synthesis the modeler would clone from the Palettes a set of bioReservoirs and bioProcesses of the appropriate types to represent the different terms of the following equations, which are then transferred to the appropriate compartments, connected and configured, and desired values are given to the model parameters. Translocation rates between compartments could be expressed as translocation half-times, defined as : $t_{1/2.nu} = \ln(2/K_{nu})$, where K_{nu} is the rate constant for processing in the nucleus or the rate constant for transport from the nucleus to the ER, or both lumped together. For steady-state balanced growth, where $\mu=0$ and rp(t) is the rate of transcription in the nucleus (molecules/hour/cell): $(RNA)_{nu.ss}(t) = rp(t)/K_{nu}$. The additional equations that follow are similar to those proposed for cell cycle-dependent protein accumulation by Bibila & Flickinger (1991) Biotechnology and Bioengineering 38: 767-780, and have to be adapted for use in the simulation formulas of the system of this invention which takes a dynamic approach:

$$[X_1] = \S X_1 * [X_1]_o * e^{-k1*t} \text{ (this is ER)}$$

$$[X_2] = \S X_2/\S X_1 * [X_1]_o * (e^{-k1*t'} - e^{-k2*t'}) * k_1/(k_2-k_1),$$

where $t'=t-d_1$ (this is Golgi)

$$[X_3] = \S X_2/\S X_1 * [X_1]_o * (-(e^{-k1*t''}-1)/k_1 + (e^{-k2*t''}-1)/k_1)/(k_2-k_1),$$

where $t''=t-d_1-d_2$ (this is extracell. comp. for secreted proteins)

[0170] On many occasions, the newly generated molecules are secreted, while on other occasions, the newly generated molecules are retained in the cell, either to maintain housekeeping functions which are usually included in the steady-state and no implicitly modeled, or they result in new cellular functions in which case its mechanisms are graphically modeled. If this response results only in proliferation, the inference engine may monitor the simulation of the cell mechanisms (FIG.28), and based on the values of certain variable or sets of variables, may cyclically activate (2802) the next phase-compartment in the cycle. However, if the response results in cellular differentiation, the inference engine activates (2805) a different cell-bioModel, if such exits, that represents the new stage of differentiation, or will send a message to the user otherwise.

[0171] The different bioProcesses in a cellular bioModel may be activated for different periods of time, if placed within sequential time compartments (2421), but they will be overlapping with many other bioProcesses encapsulated in the G0-phase-compartment (2404) which are not deactivated during a simulation. BioReservoirs that may be participate in two different sequential time compartments are preferably located within the G0-phase-compartment, to maintain the continuity and to carry over the values of its concentration / density / scaled-amount from one sequential discrete time compartment to the next. These temporal compartments represent somewhat defined cellular phases with different activities.

[0172] Mechanisms that are well studied in one animal species can be interpolated in the pathways of other species where the pathway may be less understood, such as in humans, by cloning the submodel (2408) encapsulateing the bioReservoirs (2413) and bioProcesses (2409) from the first species and transferring to the appropriate compartment of the second, and then renaming and reestablishing the appropriate connections.

[0173] BioPools that represent points of entry for non-physiological substances, such as drugs or synthetic agonists, antagonists or inhibitors that are not produced within the cells, at the compartment where it is first introduced have the following characteristics: a) they receive their values only from external sources, such as a simulation-panel, initially, the concentration equals the external input, and therefore • inputs = entry Value; b) they have a basalConcentration value of 0, c) they have no schematically modeled inputs, d) e) they may have several schematically modeled outputs, through modification- and translocation-bioProcesses, in addition to the decay term that represents diffusion, degradation and dilution, and e) the quantity variables of those bioPools cannot obtain values higher than those inputted by the user, but may get smaller over time due to the outputs and the decay factor ([As]*(μ+• outputs) • [entry]). When that compound is modified or translocated to another compartment, the latter's is a regular bioPool, but still constrained.

• Modeling Cell Population Dynamics

[0174] There is a number of ways of modeling and simulating cell population dynamics, using this invention. A number of attributes defined for the classes cell Reservoir, cellPool, cell-bioModel, and other bioObjects can be used to hold parameters and variables to be used for that purposes. Following the teachings of this invention, other attributes can be similarly defined and used to better meet any particular biological system to be modeled, or hypothesis that a user may want to test, or to meet the needs for any monitoring or control system that use those biological models. Some examples are presented below. The state transitions are represented by translocation bioProcesses (405, 407, 414, 417) between two bioPools of entities in the former and in the latter states, at a rate that is determined with one of three approaches: a) mechanistic approach; b) probabilistic approach; or c) deterministic approach.

[0175] In the **mechanistic approach**, the rate of the translocation process will be dependent on the dynamically changing values of certain variable or combination of variables, implicitly modeled inside an indicated cell-bioModel (FIG.24), reaching some threshold value(s). With this approach, the high-level models may be integrated with the mechanistic models. representing both the spatio-temporal integration in a single cell-bioModel and the quantitative data about populations of such cell modeled. This mechanistic approach is currently difficult to implement for the particular domain selected for this invention, since there is not sufficient information available. This approach can be implemented with better known systems in other domains.

[0176] For some simplified in vitro biological systems, such as oocyte extracts, a mechanistic approach is used to model a transition which mechanism is known or expected, such as the role of the interplay between cyclin-B2 and the phosphorylated and dephosphorylated forms of the cdc2-kinase in the transition from the G2 to the mitosis phases of the cell cycle. The rate of increase in the number of cells with a cdc2**cyclin (cc) level greater that ø in a given cell cycle phase = fraction of cells in that phase (G_1) * {the rate at which cells within that phase reach a cdc2**cyclin level ($[cc]^{G1}$) greater that ø - the rate at which cells-with cdc2**cyclin level greater that ø leaves that phase }:

[0177] The following is an example of how a simplified mechanistic approach can be implemented. The rates of growth in eukaryotes and the rates of translocation from one cell-state-pool to the next can be represented as a linear or non-linear functions of the concentrations of specific molecules, such as external regulatory factors, regulatory enzymes, and total and specific mRNAs. For example, if a limiting protein L determines the transition from one time-compartment to another, then the expression of the mRNA for L could regulate that transition. Furthermore, the growth-rate μ can be represented as a linear function of the concentration of a given specific mRNA, and the rate of mRNA transcription can be represented as a non-linear function of the concentrations of two or more growth factors (or other regulators). or it can be assumed to be a combination of growth associated and stimulus-dependent kinetics. In turn, the steady state values of all enzyme synthesis rates and the total specific RNA level are a function (linear for RNA) of the steady-state specific growth-rate μ. The total RNA can be used as a measure of the growth resources of the cells, since the ability to increase a stimulus-induced specific sRNA is limited by the amount of synthesis proteins, which is limited by the amount of total RNA. Because the relatively high rate of RNA turnover and the dilution effect of cell growth, the [sRNA] responds immediately to a removal of the stimulus-effector. With an increase of the stimulus-effector, the rate of increase of [sRNA] is controlled by a time constant which is inversely proportional to the total RNA of the cell (Copella &Dhurjati 1990. Biotech. Bioeng. 35:356-374). The modeler would clone from the Palettes set of bioReservoirs and bioProcesses of the appropriate types o represent the different terms of the following equations, which are then transferred to the appropriate compartments, connected and configured, and desired values are given to the model parameters. The following equations have to be adapted for use in the simulation formulas of the system of this invention. ($\mu = k_x$ * ([mRNA] - [mRNA]o)→ (1/h) = (gX/gRNA) * (gRNA/gX/h); $\tau = k_x/(\mu + k_x$*[mRNA]o/[X], where k_x is the rate constant for mRNA transcription (gRNA/gX), τ is the time constant (gX/gRNA), and [mRNA]o/[X] is the specific [mRNA] at μ=0. The growth rate can also be viewed as the sum of all rates in one simple anabolic pathway divided by the total mass (growth from glucose in bacteria): $\mu=(\bullet r_i)/[X]$. The number of generations: Φ = log(2)((X - Xo) /Xo)); μ * G_1* $e^{\mu t^*}$ $\bullet_{ø}$* $\lambda^{G1}(x)dx$ = G_1* $e^{\mu t}$ * ($[cc]^{G1}(ø)$ * $\lambda^{G1}(ø)$ - $r(G_1 \rightarrow S)$ * $\bullet_{ø}$* θ(x)dx), where $\lambda^{G1}(x)$ and θ(x) are the frequency functions describing the cc content distribution for cells in G_1 and $G_1 \rightarrow S$ transition, respectively, and $r(G_1 \rightarrow S)$ is the rate constant that describe that transition; $r(G_1 \rightarrow S) = \mu / G_1$* $(N_0 + G_2 + S)$; $r(S \rightarrow G_2) = \mu / S$* $(N_0 + G_2)$; $r(G_2 \rightarrow G_1) = \mu / G_2$*$N_0$

[0178] The following equations are examples of how overall rates of synthesis can be used to implement black-boxes, where each of those equations provide the value for the Velocity of the black-boxes to be represented by the corresponding type of bioProcesses: for IFN-α → IL2 : $rs_{(TF\text{-}IFN\text{-}a)} = k_1$ * [IFN-α] and $rs_{(mRNA\text{-}IL2)} = k_2$ * [TF-IFN-α] (or $rs_{(mRNA\text{-}IL2)} = 1 + k_a$ [IFN-α) / 1+ k_a [IFN-α]+ k_s;); $rs_{(IL\text{-}2)} = k_3$ * $[mRNA_{IL2}]$.

[0179] By using the concentrations variables of sol-mol-pools, expressed as moles per volume units rather that per cell, it would not be necessary to adjust the steady-state concentration with growth for constitutive enzymes, since it is assumed that both increase at the same rate, while for regulated enzymes, the dilution effect is more apparent, since it is not paralleled by synthesis. In the current implementation focusing on regulatory processes, since most of them take place in the form of large molecular complexes, highly localized in some parts of the cell, it would be more meaningful to use the defined density variables (molecules per cell). To synchronize the simulation from the intracellular to the cellPool level, the quantities of all the bioPools of extracellular molecules are halved at each cell division transition. In addition, specific rates are rates per cell, and total rates are specific rates multiplied by the cell density. Specific cell components may increase throughout the cell cycle: a) continuously at a constant rate, b) at a rate that doubles when the DNA is replicated, c) at a progressively increasing rate which follows fist-order kinetics. or d) discontinuously, according to a sequence of events which changes the rates in a complex pattern.

[0180] When the mechanisms are unknown, a **deterministic approach** may be taken with the transitions modeled as a time-dependent events. As shown in Fig 3, the cell-phase compartments, which are holders of mechanistic models, also have an attribute Interval (317) to hold information about the experimentally observed duration of that phase of the cycle, which can be used during a simulation-run to deterministically deactivate such subworkspace when that period of time has elapsed since the subworkspace was activated. or whenever any type of known constraints are met (FIG.28). In the deterministic approach, the rate of the translocation process is defined by a rate constant, which may have a constant predefined value, or its value may be variable and set at run-time by a simulation formula, an expert rule, or procedure, such as one that reads the phase's "interval" just described, or it can be modeled by a model-block. The transfer of cells from cellPool i to cellPool j does not happen at once, but rather follows a Gaussian distribution, which may be modeled using the corresponding model-block.

[0181] Following is an example of a high-level representation of the effect of some drug on cell growth, when there is partial mechanistic knowledge available, but where most of the many other steps in the pathways leading from the external drug to the regulation of cell growth are unknown. In a system where enzyme X_1 produces a soluble substance c which is used in the formation of active enzyme X_2. If $\mathbf{X_1}$ and $\mathbf{X_2}$ represent the total amounts of X_1 and X_2 contained in n cells, and if X_2 is the component which, upon reaching a threshold, determines the occurrence of cell division, so that $n = \beta X_2$, where β is a constant, then, assuming steady-state, when c remains at a steady concentration and therefore the formation of X_2 is proportional to $[X_1]$), follows that: $d[c]/dt = A(\mathbf{X_1}/n) - B(\mathbf{X_2}/n)[c] - C[c] = 0$, (1) where A, B, and C are constants. From (1) it can be said that $c = \beta\ \mathbf{X_1}/\mathbf{X_2}$, and following a set of derivations it results that the growth rate is: $k = d(\ln n)/dt = d(\ln \mathbf{X_2})/dt$. If now c decreases from the action of some drug, then $\mathbf{X_2}$ which is proportional to [c] decreases. The cell growth rate $dn/dt = \beta\ d\mathbf{X_2}/dt$ will also decrease. The effect of the drop in c is an increase in $\mathbf{X_1}$ relative to $\mathbf{X_2}$, until a new level of $\mathbf{X_1}$ is reached that restores the cell growth rate to its initial value. In the representation used in this invention:

- the 1st term is represented by an enzyme-bioProcess corresponding to the formation of c by X_1, where the enzyme-bioReactant would be connected to the bioPool of X_1, the substrate-bioReactant would be connected to the bioPool of the precursor of c, and the bioProduct would be connected to the bioPool of c;
- the 2nd term is represented by an complex-formation-bioProcess corresponding to the consumption of c in the formation of X_2 from c and other subunit(s), connected to the appropriate bioReservoirs; and
- the 3rd term could be represented, depending on the specific biologic mechanism to be modeled, by: a) a complex-formation-bioProcess corresponding to the consumption of c in the formation of a complex leading to the loss of c by degradation by a ubiquitin-dependent mechanism, connected to the appropriate bioReservoirs; or b) a translocation-bioProcess, representing other mechanisms of loss of c, such as by diffusion, connected to the appropriate bioReservoirs: or c) implicitly represented by the degradation-coeff within the formula that determines the current amount of c, which includes a term with default degradation.
- in this system, the scaled-amount set of variables is used, because what is relevant for the behavior of the system are the relative value of $\mathbf{X_2}$ in relation to its threshold, which determines cell division, and the ratio between $\mathbf{X_1}/\mathbf{X_2}$, which regulates the rate of production of X_2, and not their absolute amounts.
- the value of the number-of-cells attribute of the cell-bioModel that encapsulate those bioProcesses and their corresponding bioReservoirs, is doubled when the scaled-amount of the bioPool of X_2 reaches the threshold, and at the same time, $\mathbf{X_1}$, $\mathbf{X_2}$ are halved while [c] remains constant.

[0182] The above example is an oversimplification, and in fact much more is currently known about the mechanisms that determine the initiation of mitosis, in which complex interdependencies are involved, such as the equivalent of X_2, a complex, regulating not only the rapid increase in its production by a positive feedback loop upon the equivalent of X_1, but also X_2 regulating its destruction by activating an enzyme in the ubiquitin-dependent mechanism, and X_2 regulating its inactivation by activating another enzyme that converts X_1 back to c. In general, in the absence of diffusion, the simple case that might produce organization in time is that of two concentration variables, X_1 and X_1, where the value of each is dependent on the value of both, such as when: $d[X_1]/dt = v_1\ (X_1,X_2)$; and $d[X_2]/dt = v_2\ (X_1,X_2)$.

[0183] Other high-level physiological events can be modeled in a similar manner, such as considering different cells subsets (i.e. neutrophils) changing compartments (intravascular → interstitial space during inflammation) as induced by cytokines (i.e. TNF-α, which is produced upon activation by macrophages, neutrophils, T-lymphocytes, natural killer cells and mast cells), in chemotactic-bioProcesses. The time-period that a cell subset stays in such a compartment depends in part on the expression of adhesion proteins (such as E-selectin (ELAM-1) or ICAM-1 by endothelial cells, which expression is induced by TNF-α), that will be components of cell-interaction-bioProcesses.

• Modeling Progression through Different Cell States: The Cell Cycle and Cell Differentiation

[0184] Each cell-phase and cell-bioModel has two attributes called the Generation-number (318, Φ) and the Number-of-cells (319, N, per volume unit). At each cell division at the mechanistic level, that is, when the subworkspace of the

M-compartment is deactivated and optionally the subworkspace of the G1-compartment is activated, predefined expert rules (Table 10) set $\Phi = \Phi + 1$ and N=2n for each cell-phase of that cell-bioModel. Another attribute called Mass-per-cell (X), can optionally be given a value and used to calculate specific values as a function of cell mass, in which case predefined expert rules (Table 10) set X = X/2 for the Go-phase. and the G1-phase if applicable, of that cell-bioModel, and then the value of that attribute would be modeled as dependent on time or as dependent on any other variable(s), such as the concentration of some specific mRNA(s) or protein(s), until reaching again a value of X or close to X. In turn, the value of X can be used to adjust other quantities or to control the activation and deactivation of the subworkspaces of each successive cell-phase of that cell-bioModel, making it dependent on cell-size. The equivalent at the population level at mitosis is that n cells from the M-phase bioPool are translocated to the G0-phase (or the G1-phase depending on the situation) bioPool while multiplied by 2, so that the C0(G1)-phase bioPool receives 2n cells.

[0185] FIG.28 is a representation of the mechanistic approach and attempts to represent on paper what can be achieved dynamically at run-time applying the innovations of this invention. If we assume that a cell-is in Go-phase state and focus on that state's sub-workspace (2801), and there we assume that variables represented as Z and W have received the values z and w, respectively, then a rule will be invoked (2802) that will activate the subworkspace of the G1.1-phase (2803) compartment. Activation of G1.1-phase.sw implies that all the variables and parameters encapsulated within the bioObjects contained upon that subworkspace will be available to the simulator, and the data flow can now continue through any branch of the pathways represented there. Therefore, upon required events being satisfied, such as Z and W reaching certain thresholds -which could be an effect of binding one or more growth factors to their receptors on the cell membrane, the cell makes a transition from the G0 to the G1.1 phase. The subworkspace of the G0-phase remains activated during the whole simulation run because it encapsulates all those bioProcesses that are not cell cycle specific. Focusing now on the simulation of the G1.1-phase (2803), two expert rules could be invoked. saying that "Whenever variable A receives a value a > ø1 then activate G1.2-phase.sw" (2804), or "If X and Y receive values x > ø2 and y > ø3 then activate DifferentiatedCell.sw" (2805) and depending on which one is invoked first, either the subworkspace of the G1.2-phase (2806) compartment or the subworkspace of the Differentiated-Cell. sw (2807) compartment will be activated, since the subworkspace of the G1.1-phase (2804) compartment will be deactivated by another two expert rules that state that "Whenever the subworkspace of the G1.2-phase (or Differentiated-Cell) is activated then deactivate the subworkspace of the G1.1-phase". This way of reasoning apply to the other cell-phases as well, with the difference that the options vary for each cell-phase. For example, the G1.2-phase (2806) and S-phase (2808) may be committed to only one option, with the time for progression to the next phase being determined by one rule, as shown, while the G2-phase (2809) and M-phase (2810) have two options, and which one will be activated is controlled by two rules. One of the options, Apoptosis (2811) is a dead-end stage, representing regulated cell death. While in M-phase (2810), a live cell's program tells it, before or during cell division, to continue proliferating, in which case one or both the daughter cells entering again the G1.1-phase, by activating the subworkspace of the G1.1-phase, or to exit the cell-cycle, in which case the simulation continues with only the Go-phase. Note that the options for each one of the cell-phases have been predefined by the characteristic number of stubs for each subclass.

[0186] The dynamics system diagram (2812) in that figure is used to represent that the defined attributes of the subclasses of cell-phase contain a variable that allows to keep track of the proportion of the cells, from the original number on Go-phase, are progressing through the phases of the cycle, as an alternative to build a model with a set of bioReservoirs and bioProcesses. This variable can be modeled using a probabilistic or mechanistic approach. In a probabilistic approach for representing cell-phase distributions, a balanced growth type of cell population is characterized by such distributions, which is a time invariant property even if their total number increase exponentially with a constant specific growth rate μ. Each of the subsets of that population, which represent different cell-phases, and which can also be represented as separate cellPools in the system of this invention, increase in number at the same specific rate and are also characterized by time invariant property distributions, which is represented by the Typical-fraction (320) attribute of each of the cell-phases. At some steps during a simulation, the alternative cell-phase compartments that could follow may exclude each other at the single cell level, but not at the cell population level, so that different subsets of a population may follow different paths, with the distribution depending on the strength of signaling events generated within the simulation itself. Therefore, the simulation of the value of this variable could be modeled to represent the probabilistic distributions, independently of which subworkspaces are activated first following the mechanistic approach.

[0187] As shown in FIG.29, If a cellular response results only in proliferation, the simulation of the population cell dynamics involving bioPools of cells in different states, results in cells that keep translocating from bioPool to bioPool, each representing a different one from a discrete number of states, resulting in cycles through a closed-loop set of cell-state bioPools, one of those . However, if the response results in cellular differentiation, it generates branching to a new closed-loop set of cell-state-pools. FIG.29 also shows how a cell populations approach bioModel can be build to model the translocation of cells between the major states considered, as mechanistically represented by the defined subclasses of cell-phase, by creating and connecting a set of bioReservoirs and bioProcesses, as the example detailed in the figure shows. The initial state is the resting state, represented by the cellPool Cell-X1-Go (2901) of cells of type

X1 (or in differentiation stage X1) in the Go-phase. That cellPool may receive inputs (2902) from various sources, represented by bioProcesses in which cells are added to the resting pool after having been activated or cycling as the result of a variety of factors, and its outputs (2903) may contribute also to a variety of such bioProcesses, such as the one that will be here described. While at Go-phase, if the cell is activated (2904) by a growth factor (2905), antigen or any other activating stimulus, the cell enters the activated state or G1.1-phase (2906) and added to the corresponding bioPool (2911). From that bioPool (2911), all the cells may follow one of the two different options, or different cells may follow different options for the next cell-phase, depending on the type and strength of the signals received, either: a) following the cell cycle (2908) if certain growth factor(s) (2909) are sufficiently present, resulting in cells X1 in G1.2-phase pathway (2910) and added to their corresponding bioPool, or b) following the cell differentiation pathway (2912) if certain differentiation factor(s) (2913) are sufficiently present, resulting in cells X2 in G1.1-phase (2914) and added to their corresponding bioPool (2915). Each of the cell types will be then running in parallel, with their own set of bioReservoirs and bioProcesses. The same reasoning applies to the rest of the system, as detailed in the figure, which will not be further explained, only to say that at the end of the cell cycle, in the cellEngine (2916) that returns cells from M-phase (2917) to Go-phase (2918), has not only a translocation function, but in addition, given that the Stoichiometric-coeff of cell-X1-Go is 2 while the Stoichiometric-coeff of cell-X1-M is 1, it doubles the number of cells to be added to the cell-X1-Co-pool.

[0188] In fast growing organisms, the dilution factor resulting from cell growth is represented by μ, while in slow growing cells direct enzyme degradation is more important, since the levels of many enzymes in animal cells is controlled by the balance between enzyme synthesis and degradation, called turnover. The synthesis of an enzyme is a zero-order process, while the degradation usually has a first-order kinetics (proportional to the concentration of enzymes. It is very difficult to measure directly the rates of change of different cellular components at the single cell level. A modeling framework requires at least data from measurements about the distributions of single-cell components in growing cell populations, and at the time of cell division and birth, which in some cases can be obtained by multiparameter flow cytometry.

3. Navigation and General Modes: Exploring and Queering the Virtual Models Menus and Panels

[0189] The domain-menus associated with the Navigation Mode and the General Mode, the default user mode for users with no modeler or developer rights, comprise the headings: "Help & Mode Menus" (1601), "Structure Libraries" (1619), "Pathway Libraries" (1628), previously discussed, and "Panels", which options include panels of various types used to access tasks that are not initiated from one single bioReservoir or bioProcess, or their components, but which are either application-wide or including more that one starting points.

[0190] The Initialization Panel is used to request the initialization tasks associated with each of the buttons. Here we will discuss the methods associated with each of those buttons. An initially rule causes this panel to be automatically displayed when the program is started. Selecting the "Menu" button calls the basis-menu-head-callback which causes the creation of the specific domain-menus, by calling other procedures defined within the Shell and, depending on the current value of the user-mode of the window from which the request has been made, it passes as argument the name the menu-structures specific for that mode.

[0191] Selecting "References Scroll" displays the the subworkspace of the References Panel, discussed below. Selecting "Structure Queries" creates a clone of Master-General-Structure-Query-Panel and displays its subworkspace discussed in more detail below in relation to FIG.30. Selecting any of the "Experiment Setup #" displays the subworkspace of the corresponding "Experiment Setup Panel", which are variously modified clones of the Master-Experiment-Setup-Panel, one of which will be discussed in more detail under the Simulation Mode heading in relation to FIG.35, since part of the capabilities provided within this type of panels can be used under the General Mode, and all the capabilities can be used in Simulation Mode.

Initialization

[0192] Selecting the "**Navig Init**" button calls the navig-init-callback, which calls the navig-init-proc resulting in the activation of the subworkspaces of all bioProcesses and bioReservoirs and then successively calling the following procedures:

> the br-initialization-procsets the appropriate values of the Master-bioReservoir (1208) attribute of all bioReservoirs, highlights the flag-color region of the icon of those bioReservoirs that have Warnings, sets the appropriate values of the Ref-bioprocess (2235) attribute of all bioPool-posts of all bioReservoirs. and establishes all the appropriate "a-down-bioProcess-of" and "an-up-bioProcess-of" relations between bioProcesses and bioReservoirs, making those changes permanent (note that relations are transient in this Shell and therefore they are lost when the application is restarted).

the bp-initialization-proc changes the color of the type-color region of the icon of each bioProcess to equal that of the bioEngine it encapsulates, sets the appropriate values of the Master-bioprocess (1405) attribute of each bioProcess, highlights the flag-color region of the icon of those bioProcesses that have Warnings, sets the appropriate values of the Ref-bioreservoir (2234) attribute of all bioRole-posts of each bioProcess, establishes "an-upstream-bioReservoir-of" and "a-downstream-bioReservoir-of" relations between certain pairs of bioReservoirs that are separated only by one bioProcess, and establishes "an-upstream-bioProcess-of" and "a-downstream-bioProcess-of" relations between certain pairs of bioProcesses that are separated only by one bioReservoir.

the downstream-chaining-proc completes the downstream chaining process by extending those relations to distant bioReservoirs and bioProcesses, by establishing the appropriate "a-downstream-bioReservoir-of" relations between any bioReservoir A and each other bioReservoir that is positioned downstream of A in any pathway, and establishes all the appropriate "a-downstream-bioProcess-of" relations between any bioProcesses B and each bioProcess that is positioned downstream of B in any pathway.

the upstream-chaining-proc completes the upstream chaining process by extending those relations to distant bioReservoirs and bioProcesses, by establishing the appropriate "an-upstream-bioReservoir-of" relations between any bioReservoir A and each other bioReservoir that is positioned upstream of A in any pathway, and establishes all the appropriate "an-upstream-bioProcess-of" relations between any bioProcesses B and each bioProcess that is positioned upstream of B in any pathway.

the biomodel-initialization-proc scans each bioModel for other encapsulated bioModels, bioProcesses and bioReservoirs, going down the subworkspace hierarchy, and establishes the following relations between any combination of them: bm-contained-in between a bioModel and a bioModel, bp-contained-in between a bioProcess and a bioModel, and br-contained-in between a bioReservoir and a bioModel. Those relations are important for all further processing, and are used by the newly defined query procedures, navigation procedures, and simulation procedures described below.

[0193] A rule monitors the application at run time, and whenever the user moves a bioModel the rule starts the biomodel-ws-check-proc, which checks whether the bioModel has been moved to a different workspace, in which case it calls the remove-biomodel-containing-proc, to brake the old relations that no longer apply, and calls this biomodel-initialization-proc to establish the new relations. The upstream and downstream relations are also created and broken at run-time by a set of rules (Table 11) that monitor whenever any such chaining relation is newly established or broken, and in response to such events they establish or brake other relations, according to similar criteria, which trigger the same rules for a chain of other bioReservoirs or bioProcesses. The use of those rules makes sure that if a modeler or user deletes a bioReservoir or bioProcess that is part of one such chain of relations, then the chain is broken between the elements of both sides on the breaking point. Such sets of rules are used for establishing relations whenever new connections between bioReservoirs and bioProcesses are made by the modeler.

[0194] The "Add Query" button is to be selected only after the "Navig Init" was been successfully concluded. If changes to the database have been made by the modeler since the "Navig Init" was last concluded, or if the application has been restarted, then the **"Query Init"** button should be selected. The common purpose of both buttons is to organize knowledge about the structure of all the bioEntities in the application, and to create a set of structures to hold that knowledge, to be used any time that the user request a predefined query that involves that kind of knowledge. Selecting the "Add Query" button calls the add-query-callback, which calls the two procedures that follow. Selecting the "Query Init" button calls the query-init-callback, which calls the navig-init-proc and the two procedures that follow:

the et-initialization-proc scans the loaded application for every copy of each named complex-bioEntity, and then assigns a value to the Id attribute of each such copy that is equal to the name of such bioEntity complex-bioEntity ended by the copy number. Copies are numbered consecutively, in the order found, with the original being -0.

the init-query-tables-proc activates the subworkspace of the Query-Arrays-Bin and deletes any old array upon it, and also deletes any old list upon the Query-Lists-Bin, and then call the following procedures:

the find-bioentity-definitions-proc scans the Bioentities-Defs-Bin, which contains all the object definitions of all the subclasses of the class bioEntity, and for each of those subclasses it calls the create-query-array-proc which creates a query-array, sets its Major-class attribute equal to the name of the superior-class of such subclass, sets its Ref-component-class attribute equal to the name of such subclass, and sets its Name attribute equal to the name of such subclass followed by the ending -query-array, and transfers such query-array to the Query-Arrays-Bin at a location specific for each superior-class;

the create-query-lists-proc creates a query-list to correspond to each query-array in the Query-Arrays-Bin, and transfers the list to the corresponding location upon the Query-Lists-Bin, sets its Ref-array attribute equal to the name of such array, and sets its Ref-component-class attribute equal to the Ref-component-class attribute of such array; then, for each of those lists and for each bioEntity of the subclass named by the Ref-component-class of such list, it calls the find-named-bioentities-with-component-proc which in turn, if such bioEntity has a name it directly calls for such bioEntity the procedure that follows, otherwise it calls first the find-named-bioentity-proc to scan the hierarchy of bioEntities superior to such bioEntity, until one is found with a name, in which case, it now calls for the named bioEntity the procedure that follows:

- the seek-bioentity-copies-proc inserts the argument bioEntity in such list, and then scans the application for any copy of such bioEntity, and if any is found then it loops back for such bioEntity by calling the find-named-bioentities-with-component-proc mentioned above, which will repeat the cycle back to this procedure, as described, until the various loops for each bioEntity of every list are completed.

[0195] When this process is completed, the list for each bioEntity subclass contains, with no duplicated elements, all the named bioEntities that encapsulate one such bioEntity, either directly or by reference. To give an example, that means that if a named protein instance-A encapsulates a protein-motif instance-B, then every named protein complex-instance-C that encapsulates an unnamed copy of instance-A (which does not itself encapsulate a copy of instance-B, but rather points to its master-bioEntity, instance-A, which encapsulates instance-B) will also be included in the list of all bioEntities that encapsulate a protein-motif. After all such lists are completed, it calls the following procedures:

the complete-query-table-proc simply transfers the values of the elements from each of those lists to their corresponding arrays, by fist setting the length of the array to match the the length of the list, and then setting the text of the initial-values attribute of each array to equal a text string that mentions all elements of the list separated by commas. The list of elements of either the list or the array can be displayed by selecting the "describe" option of their menus.
the create-auxiliary-query-arrays-proc adds a few empty arrays corresponding to the no-selection options of each query panel, which are simply programming aids
and to conclude it deactivates and activates again the subworkspace of the Query-Arrays-Bin for the purpose of initializing the query-arrays upon activation, which means that the elements of the array take the values given by the set initial values.

[0196] The **"Icons Init"** button calls the init-icons-callback, which function is not essential for the functioning of this system, but which provides a very useful cosmetic function, by realigning the tracers in the subworkspaces of bioReservoirs and bioProcesses and making their connections invisible. Selection of this button is not necessary and it can be selected any time thereafter, when problems are detected. The same effect is achieved for individual bioReservoirs or bioProcesses by slightly moving and repositioning their bioPool or bioEngine, respectively, which triggers their respective icon-movement rules which also apply to repositioning the bioRole-posts any time a bioReactant or bioProduct is moved.
[0197] Selecting the **"Init All"** button calls all-init-callback which calls the all the initialization procedures mentioned above, and selecting the **"Hide"** button calls the hide-workspace-callback, which hides this Panel.

Navigation

[0198] In this invention, navigation refers to the domain-specific tasks of displaying and hiding either related bioObjects or their subworkspaces. The concept of related object include various types of associations, including some that are permanently stored in the knowledge base, as defined by either graphic or distant connections, or containment in subworkspaces at any level in the hierarchy, and some that are transient such as relations, which are established and may be broken at run-time (note that in the system of this invention it is also possible to store run-time versions or snap-shots of the application, which also include transient objects, such as relations). Navigation tasks may be performed in any mode, but Navigation Mode has that name because the behavior of certain object classes, including the classes bioReservoir, bioProcess, bioModel, and complex-bioEntity, is restricted to do just that. This means that a navigation task is automatically performed upon clicking on the instances of any such classes, instead of displaying a menu with various other options as well, and therefore all those other tasks offered by the other options are not available in this mode. Navigation tasks are usually simplified in this filing by expressions like "selection of A starts proc-B which displays C", but in reality these processes do more than that in this invention: selection of A invokes proc-B which takes as arguments A (the item) and the current window, and causes C to be either displayed or hidden on the current window, depending on the value of the Toggle-state attribute of A, and in addition, the icon appearance of A is configured, switching between its pressed or depressed configuration, respectively, and the value of the Toggle-state attribute of A is alternated between on and off. In such cases, A may be either a button or any of the bioObjects that has a subworkspace, which in this regard behave similarly.
[0199] Any navigation task that is directly performed in this mode may be performed in any other mode when selected from the menu. Selecting any bioProcess (2410) implies its "details" option which starts the details-bp-proc which

displays its subworkspace (2411), in the same way that selecting any bioReservoir (2414) implies its "details" option which starts the details-br-proc, resulting in its subworkspace (2415) being displayed or hidden. depending on whether the value of the Toggle-state is hide or show. When selecting the "details" of any bioProcess that has a master-bioprocess but has no subworkspace. conditions which when combined are characteristic of a copy bioProcess used in the creation of Pathways, causes such first procedure to call the bp-master-details-proc which in this case displays or hides the subworkspace of the original (master) bioProcess, in addition to configuring the icons of both the original and the copy bioProcess to their pressed or depressed appearance. In a similar way, when selecting the "details" of any bioReservoir that has a master-bioReservoir but has no subworkspace, causes the first procedure to call the br-master-details-proc. BioEngines do not have subworkspaces by default, by the modeler can create one to store tabular functions. In such cases, a "show-sw" option is available, and selecting such bioEngines implies "show-sw" which starts the show-bioengine-sw-proc which displays or hides its subworkspace and toggles the configuration of its icon. The behavior of complex-bioEntities is similar, as shown in FIG.18, where selecting any bioEntity (1808, 1813, 1815, 1817, 1819, or 1821) implies selecting the "details" option which starts the et-details-proc which displays or hides its subworkspace (1809, 1810, 1816, 1818, 1820, or 1822, respectively) and toggles the configuration of its icon. The combination of this option with the go-to-sup-obj-button (1826) upon each subworkspace allows to navigate up and down in their workspace hierarchies.

[0200] Selecting any bioProduct-post (711) or any bioReactant-post (703, 707), in General, Navigation. or Simulation Modes, implies selecting its "show-br" option which starts the rolepost-show-br-proc which displays the subworkspace (714 or 705) of the bioReservoir named by their Ref-bioreservoir attribute. Similarly, selecting any bioPool-r-post (704) or any bioPool-p-post (712) implies its "show-bp" option which starts the poolpost-show-bp-proc which displays the subworkspace (702) of the bioProcess named by its Ref-bioprocess attribute. This option is only available in the menu when the bioPool-post has a name. Selecting the go-to-sup-obj-button (1825) on the subworkspaces of each bioObject displays its superior object, while selecting the hide-sup-ws-button (x) hides its workspace . The combination of those capabilities allow to follow all the distant connections through the pathways of bioReservoirs and bioProcesses, and also up and down in their workspace hierarchies.

[0201] Selecting any model-box (1218, 1236, or 1237) in General, Navigation, and Simulation Modes, implies its "show-sw" option, which starts the show-model-box-sw-proc which displays or hides its subworkspace (1221) and toggles the configuration of its icon. This option is used to access the model-block (1222) upon that subworkspace.

[0202] The "bioentity" option appears in the menus of bioReservoirs (1803), bioReactants and bioProducts (1525) only when the Ref-bioentity attribute of the bioReservoir has a value, and upon selection shows or hides the subworkspace of the bioEntity named. The Ref-bioentity attribute is defined only for the class bioReservoir and the menu options of bioReactants and bioProducts refer to that attribute when connected.

Structure Related Queries

[0203] There are two major groups of Queries: a General-Query is based only on structure and is application wide, while a BR-Query is in reference to the bioReservoir from which it was initiated. The Query Panels are built hierarchically, with each radio-button from the first Selection Panel leading to another more specific Query Panel, where now the user can compose a selection and then initiate the Query. The word Panel is frequently used in a more general sense to refer to the subworkspace of an instance of the class query-panel.

[0204] A General-Query may be requested by selecting from the "Panels" menu head the "Structure Queries" option which calls the create-general-structure-selection-panel-callback, which clones the Master-General-Structure-Query-Panel, stores the clone in the Temporary-Panel-Bin, configures its radio-box with the aid of the set-radio-box-id-proc and displays its subworkspace. The user may now use such Panel to select any radio-button from its radio-box, which calls the select-general-structure-selection-panel-callback with such button as one of its arguments, which in turn: a) gets the value of the radio-button selected. and depending on the value of the On-value attribute of such button, clones the corresponding master-panel. then calls the create-general-query-panel-proc with such value and the new panel among its arguments, which completes such panel by scanning the panel for the presence of specifically labeled radio-boxes and configuring the radio-boxes with the set-radio-box-id-proc, and b) displays the subworkspace of the new panel.

[0205] The user may now use that Query Panel to select any combination of one radio-button per radio-box and then selecting the "Query" button calls the query-general-structure-callback, which calls first the query-general-structure-proc, which scans such panel for the presence of specifically labeled radio-boxes and then calls find-specific-array-proc with such label as one of its arguments to find the value of the selected radio-button and the matching array within the Query-Arrays-Bin (those arrays are created during the Query Initialization process). An auxiliary scroll-text-list is created and, if only one radio-box is present in the Query Panel, then the elements of the corresponding array are transferred to such list. If two radio-boxes are present in the Query Panel, then two such arrays should be found, in which case the merge-two-structure-lists-proc is called, which scans those arrays to find the common elements, which are transferred to such scroll-text-list. An specific text-string is developed to include references to the selected options and to the outcome of the search. which is then given as a value to a free-text structure which is displayed as a heading for the results of the search. The callback calls then the create-general-query-output-panel-proc with such scroll-text-list and free-text among its arguments, which creates an output panel by cloning the Master-Molecular-Query-Output-Panel, then creates a scroll-area to display the elements of such scroll-text-list, and transfers to the subworkspace of such output panel both the free-text and the scroll-area listing the results of the search, which is them displayed. The user can now use such scroll-area to interactively display or hide the structure of any number of bioEntities named by the options listed.

Pathway Related Queries

[0206] FIG.30 illustrates an example of one of the several types of predefined Pathway Related Queries that the user can perform by just selecting the desired buttons from the options offered in the panels (3004, 3006). All the searches generated from this type of panels include bioReservoirs located anywhere within the pathways that are either upstream or downstream, depending on the selection, of such initial bioReservoir (3001), and also the bioEntities referred to by any of those bioReservoirs. Those upstream or downstream queries can be further restricted by the type of roles that those bioReservoirs play in different bioProcesses, as defined by the major classes of bioReactants to which their bioPools are distantly connected, which is referred here as function-related search. In complex queries of mixed type, such bioEntities can be further restricted to those encapsulateing any particular component, similar to the Structure Related Queries described above. In such cases, the methods pertinent to the pathway search are applied first to generate the pathway-related set, followed by the function search, which is applied only to the pathway-related set and generates the function-related set, and finally the structure search, which is applied only to the function-related set, or to the pathway-related set if no function search has been selected, and generates the structure-related set. The structure search applies only to the bioEntities for the purpose of displaying the results. in the sense that all bioReservoirs that meet the selected pathway, and optionally function, requirements are listed in the scroll-area listing the bioReservoirs, regardless of whether they meet any of the bioEntity related criteria. This implementation is currently preferred because the Ref-bioentity of a bioReservoir is an optional attribute, and it is expected that many of the bioReservoirs will not have a corresponding defined bioEntity. This alternative provides a more complete search in terms of relative position and function of bioReservoirs, without being influenced by the implementation of an option.

[0207] These types of Query may be requested in two different ways: a) from a bioReservoir (3001) by selecting from its menu (3002) the "query panels" option (3003); or b) from the subworkspace a bioReservoir by selecting the query-tracer (Q) connected to the bioPool. A **query-tracer** is a subclass of list-tracer which selection implies selecting the tracer's "show" option. Each of those options starts the create-br-related-selection-panel-callback, which creates a clone of Master-BR-Molecular-Query-Panel, configures it, and displays its subworkspace (3004), the Pathway Query Selection Panel. Configuring this panel and any other panel that follows during the processing of br-related queries always includes to set the value of the Related-item attribute of the panel equal to the name of the bioReservoir from which the query was initiated, and this name is passed along when a new panel is originated from such panel. That is, any callback invoked from a button of a panel, takes the name of the bioReservoir from the Related-item of the panel upon which subworkspace it is located, and passes that name as an argument when calling other procedures.

[0208] The user may now use that Pathway Query Selection Panel to select any radio-button (3005) from its radio-box, which calls the select-br-related-query-panel-callback with such button as one of its arguments, which in turn: a) gets the value of the radio-button selected, and depending on the value of the On-value attribute of such button, clones the corresponding master-panel, then calls the set-br-related-query-panel-proc with such value and the new panel among its arguments, which configures and completes such panel with the set-radio-box-id-proc, and displays the subworkspace (3006) of the new panel, a Pathway Query Panel. This panel is composed of: a) the title (3022) which specifies the type of search being made from that panel and names the bioReservoir (3023) that is the point of reference for the search; b) the set of radio-buttons (3024) of the radio-box labeled "direction", defining the two possible alternatives; c) the set of radio-buttons (3025) of the radio-box labeled "function", which defines the possible alternatives available for function, which in this case we only show a small but important subset that is of high relevance to scientists; d) the set of radio-buttons (3026) of the radio-box labeled "motifs", which defines the possible alternatives available for subclasses of protein-motif. Many other possible sets of radio-buttons, such as those of the radio-box labeled "location", which defines the possible alternatives available for subcellular compartments, are not shown.

[0209] The user may now use that Pathway Query Panel to select any combination of one radio-button per radio-box. Selecting the "Upstream" option (3007) calls the upstream-query-lists-callback which calls the upstream-function-query-lists-proc. Selecting the "Downstream" option calls the downstream-query-lists-callback and then the downstream-function-query-lists-proc, which do the following: a) it first calls repeatedly the create-br-query-list-proc to create a set of two lists per each predefined type of function, one to contain names of bioReservoirs and the other the names of bioEntities; b) then, it loops over all bioReservoirs that are upstream (or downstream) and scans for their connections to certain classes of bioReactants, which define their function, and enters their name in the corresponding one of those bioReservoirs lists, for example, if the bioPool of a bioReservoir is connected to both a kinase.r and a transcription-factor.r, then the name of such bioReservoir will be added to each of the two corresponding lists, so that a bioReservoir may be named in more than one function list. but is not duplicated within one single list: and c) for each pair of one of those bioReservoirs lists and its corresponding bioEntities list, it calls the fill-br.et-list-proc which scans all the bioReservoirs in the fist list for the value of their Ref-bioentity attribute, which if available is added to the second list. All those lists, to be used by the procedures described below, are associated by a relation to the panel from which they are created, are valid only for the bioReservoir giving the related-item of that panel, and are deleted when the panel is deleted.

[0210] The user can now select any combination of one radio-button per radio-box, such as receptors (3008) and β-strand-motif (3009), and then select the "Query" button (3010) to complete the processing and display the results of the query. The processing from the "direction" button is independent and previous to the processing from the "Query" button. Selecting the "Query" button calls the query-br-related-callback, which first obtains the values of the selected radio-buttons of the "direction" (3024) and "function" (3025) radio-boxes, and then calls the get-br-function-list-proc with those values among its arguments, which are used to obtain and return the corresponding set of two lists from the bioReservoirs lists and bioEntities lists created and filled in the previous processing. Such bioReservoirs list is used to directly pass its values to the bioReservoirs scroll-area (3013) for final display, while such bioEntities list is used as the basis for the complex further processing that follows. The callback scans now the Panel to find whether there are structure related radio-boxes, such as the "motifs" radio-box in this example, in which case it calls the query-br-structure-lists-proc with those previous values and the bioEntities list among its arguments, which does a complex processing as following: a) it finds the label of such structure related radio-box(es) and calls find-specific-array-proc to find the value of the selected radio-button for such box(es), such as β-strand-motif (3009), and its matching array(s) in the Query-Arrays-Bin (as described for Structure Related Queries); b) if there is only one structure related radio-box, like in FIG.30, then it calls merge-br-function-structure-proc, or if there are two structure related radio-boxes it calls merge-br-function-structure-structure-proc, passing among its arguments the values of all selected radio-buttons, as well as the corresponding bioEntities list filled earlier, and the array(s) now found. These procedures create a new list, the bioEntities scroll-text-list, and then, taking into consideration the type and value of the arguments: a) they select a text specific for each situation, to be displayed as a title for the results, b) they call the br-fill-etl-list-proc or the the br-fill-double-etl-list-proc, which enter the common elements of both the bioEntities list and the array(s) in the scroll-text-list, and c) they return the text and the scroll-text-list to the previous procedure from which they where invoked, which returns them to the callback. The callback passes such text to a free-text structure, and finally calls the create-double-query-output-panel-proc with the bioReservoirs list mentioned, the bioEntities scroll-text-list, and their two corresponding free-texts among its arguments, which creates an output panel by cloning the Master-Molecular-Query-Output-Panel and stores it in the Temporary-Panel-Bin, then transfers the two free-texts to it, creates two scroll-areas to display the elements of each of those lists in a scrollable and interactive format, and calls the complete-br-panel-proc and complete-br-panel-proc, which configure those scroll-areas (3013, 3017) and transfers them to the subworkspace, which is them displayed (3011).

[0211] The user can now use the Pathway Query Output Panel (3011) so created to scroll through the lists of bioReservoirs (3013) and bioEntities (3017) that are the output of the query. Selecting and unselecting any option of such scroll-areas allows to interactively display or hide subworkspace of the named structure by calling the query-message-selection-proc or the query-message-unselection-proc. Each of the titles (3012) are directly related to the scroll-area below them and vary with the results of each part of the search, and they may omit naming those groups of buttons where not selection was made, or indicate that some search or part of the search was empty. The scroll-area (3013) on the top shows the bioReservoirs that meet the direction and function requirements, which are those named by the options listed, and selecting any of its options (3014) displays the subworkspace (3015) of the named bioReservoir. The scroll-area (3017) on the bottom shows the bioEntities that meet the combination of direction, function and structure requirements, which are those named by the options listed, and selecting any option (3018) displays the subworkspace (3019) of the named bioEntity. Those structures displayed can be used to navigate in any way previously described, including up and down the pathways from bioReservoirs (3015) to bioProcesses (3016) and so on, from bioReservoirs or bioReactants or bioProducts to their bioEntities as shown in FIG.18, or up and down the workspace hierarchies, by combining the variety of tools and methods previously described. As before, all the Pathway Query Selection Panels, Pathway Query Panels, and Pathway Query Output Panels are dynamically created and transient, and when the search is completed they are no longer needed and they are deleted, as well as their associated lists, by selecting the "Delete" button (3027), which calls the delete-query-panel-callback.

[0212] More complex queries can be performed with an additional Master-BR-Molecular-Query-Panel with four additional options, to include queries previously described in combination with an additional constraint criteria, the location of the bioReservoirs selected by the previous criteria within the different cell compartments. Selection of any of those new options displays the corresponding Query Panel, and in this new set of Panels, the "Downstream" and "Upstream" radio buttons are associated instead with the downstream-function-location-callback and upstream-function-location-callback, respectively, which call the downstream-location-query-lists-proc or its upstream equivalent. Those procedures call some of the procedures previously described and also call the find-compartment-proc. The user may additionally select any combination of one radio-button each from the radio-boxes offered in each particular panel, and then selecting the "Query" button now calls the query-br-related-location-callback, which depending on the boxes available on the particular Panel calls one of an alternative set of procedures, which call other procedures. Such complex processing uses a similar approach and methods as those previous described to create a new set of lists and to merge lists with bioReservoirs meeting the different criteria.

[0213] The underlaying shell loads the desired application modules on memory for run time operations, but it is able to merge and remove modules while the system is running. Since the inference engine's reasoning and search space is limited to those modules that are loaded, it is necessary to create a mechanism to allow for searches in modules that are not yet loaded on memory. This is accomplished by maintaining in the repository module an object, called the query-arrays-bin, which holds in its subworkspace a set of item-arrays for intelligent dynamic merging and removal of the modules required at particular times, particularly when searching through the bioEntity-module and in order to allow for searches in a modularized application. Those contained-in-module arrays are built following similar methods as those described here for bioModels. The difference is that a new set of relation is defined, of the series -contained-in-module, and an initialization procedure that loops over all the workspaces of each loaded module, finding out the value of their assigned-module, and then looping over all the bioModel Libraries upon each workspace, and over all the bioModels upon each bioModel Library, and then continuing with the methods described in the previous section.

Interactive Lists and Pathways

[0214] In addition to the various query methods previously described, this invention provides several other tools and methods to enable the user to extract the knowledge build entered into the system by the modeler, in addition to the new knowledge created by applying the methods of this invention to analyze and integrate the knowledge provided by the modeler. One of those capabilities is provided by the "list copies" option (1825) defined for the class complex-bioEntity, which upon selection calls the et-list-copies-callback, which creates a scroll-text-list and then calls the list-local-bioentities-copies-proc, which scans all the bioEntities of the same major subclass as the one bioEntity from which the request originated to find if the master-bioentity of any of then matches the name of such bioEntity, in which case the value of the Id attribute of each of those copies is added to the list. If there is any element in that list, then the callback creates an Output Panel by cloning the Master-Local-Bioentity-Copy-Panel, creates a specific title and a scroll-area, which are then added to the Panel and configured, and finally the Panel is displayed. This Panel looks like the Query Output Panel (3017) discussed above, with a different title. Selecting and unselecting any option of such scroll-area call et-copies-message-selection-proc or the et-copies-message-unselection-proc which interactively displays or hides the subworkspace of the structures referred to by their Id, rather than by name. This also provides an example of how, in general, instances of bioEntities as well as bioReservoirs, bioProcesses and bioModels,or any other structure, can be referred to by the value of any attribute. rather than by their names, which have to be unique and sometimes inconvenient, but they can be avoided. Referring to attributes, defined for such or any other purpose, also allows to refer to groups of instances within a class, without having to refer to each individual member of the group.

[0215] Another novel teaching of this invention is the capability of creating lists (3107) of downstream and upstream bioReservoirs and bioProcesses. By selecting the icon (3208) of any of those lists, the user can display on a workspace (3209) the elements (3221) of that list, which are pointers to all the bioReservoirs (3210) or bioProcesses that meet the criteria set for each specific list, and each pointer further-allows to either display the table of attributes of such object, or displays the object itself upon its workspace (3210). Navigation can be now initiated from those objects as well as any other tasks described in this invention. This capability is accessible through the "show lists" options from the menus of bioReservoirs (3220) and bioProcesses (3103), which appear in the menus only when they are named, as well as from the list-tracers in their subworkspaces.

[0216] As illustrated in FIG.31, chaining-tracers are iconic programmed objects connected to a bioPool (L) or to a bioEngine (L, 3105), where they are used to control the dynamic creation and display at run-time of a set of such lists of either bioReservoirs or bioProcesses that are either upstream or downstream of the bioReservoir upon which subworkspace such list-tracer is located. Subclasses include BR-chaining-tracer and BP-chaining-tracer. Selecting any such list-tracer by the user implies "show" which starts the list-tracer-handler-proc, which configures the buttons and depending on the subclass of list-tracor calls either the BR-chaining-proc (also started upon selection of the "show lists" (3220) option of the bioReservoir), or the BP-chaining-proc (also started upon selection of the "show lists" (3103) option of the bioProcess). The major actions performed by both of these procedures comprise: a) if the list-tracer already has a subworkspace, then it is displayed with its contents, otherwise a subworkspace for the BR-chaining-tracer or BP-chaining-tracer, respectively, is created by cloning the subworkspace of the prebuilt master structures BR-Chaining-Lists or BP-Chaining-Lists, respectively, which contain two sets of four predefined lists each, one set (3107) is meant to be used in any user mode, while the second set (3108) is to be used in simulation mode and contain elements from the first set that have been activated as a result of their inclusion in a simulation model, and then such lists are displayed; and b) those lists are populated having as point of reference the bioReservoir (3201) or bioProcess (3101) from which they were requested, by using for each list of the set the following criteria: the elements of DBPL and SDBPL are downstream bioProcesses, the elements of UBPL and SUBPL are upstream bioProcesses, the elements of DBRL and SDBRL are downstream bioReservoirs, and the elements of UBRL and UDBRL are upstream bioReservoirs.

[0217] A further novel teaching of this invention is the capability to dynamically generate displays of interactive pathways, where these pathways comprise a chain of downstream, or upstream, bioReservoirs or bioProcesses, or both, orderly arranged in a graphic format with connections drawn between them to indicate those that are adjacent. By visually following those connections, the user can follow the pathways that indicate a dependency relationship between those bioReservoirs or bioProcesses or both. The capability to display interactive downstream pathways is accessible from several different types of structures: a) from an individual bioProcess (3116), in which case such bioProcess is the point of reference: b) from a Navigation Panel, in which case a bioReservoir is the point of reference; and c) from an Experiment Panel, in which case one or more bioReservoirs are the points of reference.

[0218] As illustrated in FIG.31, now we will refer to accessing such capability from an individual bioProcess, or from icons in its subworkspace that have such specific functions, to display either upstream or downstream pathways. Up-path-tracer (3124) and down-path-tracer (3117) are iconic objects located upon the subworkspace (3104) of a bioProcess (3101) used to control the dynamic creation at run-time of a graphic display of such interactive upstream (3110) or downstream (3118) pathway, respectively. The "show" option starts the path-tracer-handler-proc, which depending on the subclass of path-tracer calls either the draw-up-pathway-proc (also started from the "up pathway" option 3109 from the bioProcess) or the draw-down-pathway-proc (also started from the "down pathway" option (3116) of the bioProcess). The major actions performed by each of these procedures comprise:

a) it creates a subworkspace here called Upstream Pathway Display (3110) for the up-path-tracer (3124) by cloning the subworkspace of Master-Up-Pathway, which contains a set of prebuilt auxiliary structures. The text of the title (3111) is changed to refer to such bioProcess. The x-pos and y-pos (3112) are two integer-parameters used to keep track of the most distant bioProcess added to the Display and which: a) are used by the program to compute the position of the next bioProcess to be added, after which their values are updated, ii) are used by the program to be compared to defined threshold values that, when reached as the display dynamically grows, trigger a reduction in the size of the display, and iii) may be used also by the user to read the size of the Display, which may in occasion get very large, by observing their value as displayed in digital form. The set of resize buttons (3113) allow the user to resize the Display at will; b) it creates a copy of such bioProcess, by cloning it and deleting its workspace, and then transfers the copy to the Pathway Display at a specified position (3114), and displays the Pathway Display on the window from which it was invoked; and c) it calls the create-local-up-bp-proc with such original bioProcess and its copy among its arguments, which for each bioReservoir that is an-up-bioReservoir-of the original bioProcess finds each up bioProcess that is an-up-bioProcess-of such bioReservoir and then for each such up bioProcess: i) if a copy of such up bioProcess already exists upon the Display then it proceeds to iv, otherwise it creates a copy of such up (or down) bioProcess, by cloning it and deleting its workspace, ii) it computes a new value for x-pos and y-pos, iii) it transfers the copy to the Pathway Display at the new specified position (3115), iv) it creates a connection between random locations at the icon of such new (or existing) copy and the icon of the copy that is an argument of this invocation of the procedure, and v) it calls itself, but now with such up bioProcess and its copy among its arguments. Each of these procedures keeps looping and calling itself until all upstream bioProcesses have been processed, and their copies added to the Display and connected. The an-up-bioReservoir-of and a-down-bioProcess-of, as well as the a-down-bioReservoir-of and an-up-bioProcess-of, are inverse relations which were established during initialization between appropriate pairs of a bioReservoir and a bioProcess each. A similar set of procedures is used to display the downstream pathways from the down-path-tracer.

[0219] FIG.31 illustrates short examples of Upstream Pathway Display (3110) and Downstream Pathway Display (3118). The connections are directional, with the directions of the connections above the bioProcesses indicating the direction of material flows, that is from the one that is upstream to the one that is downstream in relation to each other. The directions of the connections below the bioProcesses always show the right to left direction as a result of the next copy being one that already existed in the display, such as those connections that revert to 3119 from other posterior copies. Those pathways are interactive, and selecting the "details" option of any of the icons, or changing to Navigation Mode and selecting the icons (3119, 3121), displays the subworkspaces (3120, 3122) of the originals of those copies, which allow further access to any of the navigation (3123) and all other capabilities associated with their iconic components, and allowing further access to possibly distant parts of the application. Pathways with only bioReservoirs or pathways with alternating bioReservoirs and bioProcesses can be created with similar methods, using only minor modifications. In this invention the pathways with bioProcesses are preferred because they provide the most information about the interactions within the space constrains, while the connected bioReservoirs can be accessed by just clicking on two icons. Mixed pathways may become very large.

Navigation Panel

[0220] An interface and methods are provided to constraint all the capabilities provided in regard to navigation and display of pathways, to limit the scope of the search to an space selected by the user based on the different levels of compartmentalization. This feature can be accessed through: a) a Navigation Panel, which can be requested from each bioReservoir and is dynamically created with the bioReservoir as the reference point; or b) an Experiment Panel. The Navigation Panel is to be used in General Mode, and is not available in Simulation Mode, since the methods do not activate the subworkspaces of the selected structures, as do those associated with the Simulation Mode, described below.

[0221] As shown in FIG.32, selecting the "navig-panel" option (3203) from a bioReservoir (3201) calls the create-direction-panel-callback which displays a Selection Panel (3204). Selecting now the Upstream radio button calls the select-direction-callback, which in this case further calls the create-upstream-navig-panel-callback which calls the navig-create-upstream-lists-proc, which creates two lists with all of its upstream bioReservoirs and bioProcesses, and the complete-navig-panel-callback, which creates a Navigation Panel (3205) containing a scroll-area (3206) already configured to list all the possible levels of bioModels, the compartments within the Virtual Model in which such bioReservoir or all of its upstream bioProcesses are encapsulated. The user now has to select the desired scope of the search space by selecting one of those bioModels. By selecting for example in this case the cytosol of the G0 phase of a fibroblast (3211) and then selecting the LIST button (3207) calls the list-navigation-UBP-callback which creates new lists and calls the navig-read-up-scroll-proc to populate the new lists, displayed on the Panel (3208), with the members of the previous upstream lists that are encapsulated in such bioModel and all its encapsulated compartments. Those lists display the number of elements it contains, alerting the user about the potential size of the resulting pathway, and if too large or too small the operation can be reset and a different bioModel, lower or higher in the hierarchy, can be selected. Selecting any of those lists (3208) displays (3209) its members, and selecting any of those members (3221) allows direct access to the named bioObject (3210) on its original bioModel (3211), and displaying its details (3212) allows to further interactively navigate through the upstream or downstream pathways of such particular bioObject.

[0222] The user can now select the PATHWAY button (3213), which calls the navig-draw-UBP-callback which calls the create-navig-path-tracer-proc and navig-create-local-UBP-proc to create a navig-path-tracer (3214), by cloning a master structure which subworkspace is partially built, which is transferred to the Panel (3205) and its subworkspace (3215) is displayed and further configured by calling the navig-create-local-UBP-callback, which actions are similar to those previously discussed in reference to FIG.31, resulting in the display of the selected pathway. Selecting any of those bioProcesses (3216) and displaying its details (3217) allows to further interactively navigate through the upstream or downstream pathways of such particular bioProcess. Selecting the RESET button (3218) calls the reset-navigation-callback which destroys the lists and the navig-path-tracer upon the Panel, enabling the selection of a different bioModel and repeat the process, and selecting the DELETE button (3219) deletes or recycles the Panel.

4. Simulation Mode: Simulating the Virtual Models

Initialization

[0223] The domain-menu comprises the heads "Pathway Libraries" and " Panels", and allows also changing between user modes. Before a simulation can be run, the application must be appropriately initialized for such purpose. This is accomplished by accessing the "Panels" menu and selecting the "Initialization Panel" option, which displays the Initialization Panel.

[0224] The "Add Simul" button is to be selected only after the "Navig Init" was been concluded. If changes to the database have been made by the modeler since the "Navig Init" was last concluded, or if the application has been restarted, then the **"Simul Init"** button should be selected, since it comprises the methods of the other two. The common purpose of both buttons is to scan the application for the availability of certain values of the bioReservoirs that define the initial conditions required by the simulation, and if those values have not been provided by the modeler, to infer those values from other values provided by the modeler, and if none of the alternative source of values are available, either default values will be used, including symbolic approximations, or, depending on the attribute which value is missing, to send messages to the user to provide any of the alternative values. Selecting the "Add Simul" button calls the add-simul-callback, which calls the simul-init-proc, while selecting the "Simul Init" button calls the simul-init-callback which differs in that it first calls the navig-init-proc before calling the simul-init-proc, which does the following: a) it calls the send-no-input-messages-proc which scans the loaded application for every bioReservoir that has a name but that has no inputs, either in the form of bioProcesses connected to its biopool-p-posts, or as input model-blocks connected to its bioPool, and it sends a message to the user for each of those bioReservoirs found with no inputs, as a warning for the user to take appropriated measures if those bioReservoirs are to be involved in a simulation. The simulation proceeds also without those values, since inputs can also be provided through the Simulation Panel as user inputs, or otherwise the quantity of such bioPool will initially be whatever value the modeler has set or the program will set for the basal-quantity parameter used, or its default value, and it may or may not be exhausted during the simulation run; and b) it creates a set of four lists, which are transferred to the Initialization Panel, provided to inform the user about the source of the values for the Scaling-Density (1307) attribute of each bioPool, by adding the name of each bioReservoir to one of those lists that meet the following criteria: i) the list labeled UNList contains all bioReservoirs which initial-value of the Scaling-Density was provided by the modeler; ii) the list labeled ANList contains all bioReservoirs which initial-value of the Scaling-Density was set by the program to equal the value, provided by the modeler, of the if-scaling-amount of that bioReservoir; iii) the list labeled RNList contains all bioReservoirs which initial-value of the Scaling-Density was set by the program to equal the value of the if-scaling-amount (1210) of a bioReservoir which name was entered by the modeler in the If-scaling-bioReservoir (1211) attribute of the original bioReservoir; and iv) the list labeled DNList contains all bioReservoirs which initial-value of the Scaling-Density was was set by the program to equal the default value of 100.0. To do so, for each bioReservoir the proc scans the value of its Scaling-Density and if it has a non-default value it enters the name of such bioReservoir to the first of those lists, otherwise it calls the set-scaling-density-proc which sorts and adds the bioReservoir to one of the other three lists, following the criteria mentioned above, after setting the value of the Scaling-Density of its bioPool accordingly.

[0225] The set-scaling-density-proc discussed above is the method used in the current implementation of this invention in conjunction with the scaled reasoning used as default. An alternative method for setting the values of the scaling-density attribute, to be used when more quantitative information is available for the system to be modeled if the modeler prefer the mixed absolute/scaled type of reasoning, is the.long-set-scaling-density-proc which is much more complex but also more specific, and involves the known values of the kinetic-parameters of the participating bioReactants for each process. A mixed approach can also be developed where if those kinetic-parameters are known then the reasoning of this procedure is applied, and if they are not available, then the previous more generic procedure is called.

[0226] With the one alternative way of simulation where the original bioObjects are used, when selecting Simulation Mode from another mode starts the version of the change-mode-proc which causes the subworkspaces of all bioReservoirs and all bioProcesses to be deactivated, and during the processing involved in activating a submodel for simulation, the subworkspaces of only the bioReservoirs and bioProcesses selected to participate in such simulation will be activated; while when leaving Simulation Mode to another mode causes the subworkspaces of all bioReservoirs and all bioProcesses to be activated again. The version of the change-mode-proc used with the other simulation alternative, where clones of only those bioObjects to be used are created, does nor contain the lines refering to activation or deactivation of subworkspaces.

Simulation Panel and Procedures

[0227] Because the applications build with this system can become very large, it is important to have control over which parts of the Virtual Model are included in a given simulation. To control performance, two implementations are provided with this system, each having advantages over the other depending on the hardware facilities available:

[0228] In the first implementation, the constraining of the simulation space is achieved by using activatable subworkspaces for composite bioObjects, such as bioReservoirs, bioProcesses, and certain time-compartments, and by having all those structures initialized to a deactivated state. Only the subworkspaces of the desired structures at any given time are activated, and only their contents are therefore participating in a given simulation and associated inferences. Methods associated with buttons on the entry-panels used for simulations and experiments, activate the subworkspaces of bioObjects connected along the desired pathways and encapsulated in the selected bioModel or compartment, and this activation is necessary for propagating values along the pathways. Inactivation of those subworkspaces occur upon resetting the selection within those panels. This mechanism is also very utile in activating or deactivating entire branches of a simulation model at run-time. The efficiency of large scale simulation applications is further improved by having the initiation of the simulation of different subcomponents of the high-level bioModel be driven by events, when appropriate, or by time intervals. This implementation allows only a single user to run simulations concurrently, since the attribute values of the bioObjects change during the simulation, as well as the activation state of such subworkspaces. For multiple users, multiple processes of the program have to be started.

[0229] A second implementation is based in dynamically cloning those bioReservoirs and bioProcesses, or certain time-compartments, required by the simulation, and to use those clones to hold the changing attribute values, rather than the originals. In this implementation, the subworkspaces of bioReservoirs and bioProcesses are not activatable, remaining always activated. On the other hand, the subworkspaces of certain time-compartments are still activatable, are deactivated when created, and their activation is controlled during the simulation, operating as in the first implementation. This implementation allows multiple users to run simulations concurrently, each on a different window, but increases the burden on the system by increasing the required RAM to hold the additional transient copies and to run more than one simulation concurrently. Since the system of the current invention is modularized, and the modules can be dynamically up-loaded and down-loaded, part of this problem can be solved by up-loading the module(s) that contain the originals, creating the copies for the various simulations on the top module, and removing the modules that are no longer required from the memory, before the simulation is started.

[0230] A simulation can be initiated from any named and connected bioReservoir in the Virtual Model, from the bioPool of such bioReservoirs, or from the bioReactants and bioProducts connected to such bioPools. As shown in FIG.33, selecting the "simul panel" option (3303) from the desired bioReservoir (3301), which is only available when in Simulation Mode, calls the create-input-type-proc which creates one of two types of entry-panel by cloning one of the prebuilt master structures and displays the subworkspace (3304), called a Simulation Selection Panel, depending on whether the subclass of bioPool involved represents soluble or bound bioEntities (Concentrations or Density inputs, respectively). Either Panel (3304) prompts the user to select the type of input, and the choices offered are combinations of absolute or scaled values with one-time or periodic inputs. Selection of one of the radio-buttons calls the select-input-panel-callback, which: a) according to the user's selection it calls one of a set of specific procedures, including such called in this example, the create-periodic-relative-input-panel-proc, which creates and partially configures one entry-panel of the specified type by cloning one of the several prebuilt masters; b) it creates two lists and then calls the fill-br-downstream-lists-proc to search the application for all bioReservoirs that are downstream of such reference bioReservoir, as defined by the relations established during the initial initialization, and fills the DBR list with those bioReservoirs, and then do the same for the downstream bioProcesses to be added to the DBP list, and uses those values to configure the scroll-area; and c) it calls complete-input-panel-proc to complete that panel and display its subworkspace (3305), called here a Simulation Panel. Each of the entry-panels is related to the bioReservoir (3301), or the bioReservoir referred by the bioReactant, from which the request was originated, that is, the Related-item attribute of the entry-panel newly created is a pointer to such bioReservoir. The difference between the one-time or periodic inputs typos of Simulation Panel is that the latter has two additional entry-boxes to allow the user to enter the time interval for subsequent entries (3309) and the number of subsequent entries (3308) at such intervals of the amount indicated by the entry value (3306) and starting at the indicated time (3307). The difference between the absolute or scaled inputs types of Simulation Panel is that the entry value in the former is given in some units standardized throughout the application while the latter is a dimensionless value. The prebuilt panels contain the overall layout with the common components, while other structures specific for each bioReservoir are added dynamically at run time.

[0231] The Simulation Panel (3305) can now be used to interactively set-up and start simulations comprising a desired bioModel by selecting one from the scroll-area (3312), as described in relation to FIG.32, to allow selection of an adequate size of the model to be simulated. The parameters and optional capabilities used for the simulation at run-time are configured by using an structure upon the Simulation Panel specific for that purpose, the Shell's model-definition (3318). Its table of attributes (3319) allows the configuration of any desired attributes. The Items-belonging-to-this-model (3320) is set by the program to name the bioModel selected by the user, and the variables and parameters that form part of the simulation model are all those encapsulated in any of the bioReservoirs and bioProcesses in the pathway contained in the workspace named by such attribute. This is important when dealing with very large systems, allowing to focus the computer resources on the subsystems of interest. Further attributes are used to set the time interval (3411) between simulation cycles, the type of integration algorithm (3412) desired for the state variables; the configuration for any external simulator (3413) that can also be used to receive or provide values of the variables in this model; the name of any main procedure (3414) that is invoked once every simulation cycle, which may start or call any other procedure, and which in this case calls the model-simulation-proc; a choice to send all the values to external databases or simulators (3415) at the beginning of each cycle; an indicator (3416) of whether the simulation is not-running, running or paused, or whether is nothing to simulate or is a simulation error; and the configuration of the simulator clock (3417) to run either synchronously or as fast as possible, which is more appropriate for simulating models which actions take place over long time periods. This Panel permits the user to enter the input value (3306) to be added to the Accumulation of such bioReservoir and the time (3307) of such input. Help buttons in each section of the Simulation Panel provides instructions for the user in how to proceed, such as the one (3310) which subworkspace is shown (3311). Following those instructions, selecting the LIST button (3313) calls the list-simul-callback, which finds the previous lists and then calls the read-scroll-proc to search for the members of those lists that are encapsulated in the bioModel selected by the user, as previously discussed and fill the lists labeled DBRL (3314) with downstream bioReservoirs, the SDBPL list (3315) with downstream bioProcesses, and the the UBRL list with bioReservoirs that provide the inputs for such bioProcesses but which are not included in the first list.

[0232] As shown in FIG.34, selecting one of the bioReservoir lists (3314) displays a workspace (3401) with pointers (3402) to each of the elements of that list, which allows to examine the bioReservoir that are to be included in the simulation. Selecting any of those pointers (3402) allows the user to either display the table of attributes (not shown) of that bioReservoir or to directly display the bioReservoir, and its subworkspace (3419), upon its bioReservoirs Bin.

In a similar way, selecting the SDBPL list (3315) displays the pointers (3405) to the bioProcesses that are to be included in the simulation, which upon selection (3406) directly displays the bioProcess and its subworkspace (3425). The user can examine those lists to decide whether the selection made is appropriate before proceeding. If not satisfy the user can reset the panel, and transfer to the panel additional or different submodels, and press the "List" button again.

[0233] When a Simulation Panel is created, the ACTIVATE (3316) and START (3322) buttons are disabled to prevent the user of taking such steps prematurely. When the processing associated with the LIST (3313) button is completed, the ACTIVATE button (3316) is enabled, and when selected it calls its associated procedure specific for each type of Simulation Panel. Such procedures check that a value has been entered by the user and then call activate-simulation-proc which in turn: a) looks on the Panel for the three lists created above and calls the activate-model-proc with those lists as arguments. Upon entering the Simulation Mode the subworkspace of all bioReservoirs and bioProcesses are deactivated, so that their variables become inaccessible to the simulator. The function of this procedure is now to scan the DBRL (3314), DBPL (3315), and UBRL lists and activate the subworkspaces of each of their elements, which are the ones that will be now included in the simulation, after which returns back to the previous procedure, which scans for the Basal-Density attribute of the now accessible bioPool of each of the bioReservoirs on those DBRL and UBRL lists and if its value has been modified by the user then it adds the name of the bioReservoir to the User-BC list of the set described below, and if its value has a default value then it calls the basal-density-proc for such bioReservoir.

[0234] A simulation starts with the baseline model, after being initialized by setting the initial values of the initial amounts of the bioPools (concentrations, densities, scaled-amounts, or other quantities, depending on how the bioPools involved where defined) to be equal to the values of the corresponding basal amounts. In order to activate the system, a disturbance may be introduced by entering one or more input values through the entry-panel of the desired bioReservoir, or through the desired experiment-selections of the Experiment Panel (below). By entering different values for those variables, or by changing the values of the constant parameters, what-if analysis can be performed on the system. Montecarlo simulations can be performed using a montecarlo-model-block as an input to each of the bioPools desired as test inputs, and the simulation can then be started from either the entry-panel of a desired bioReservoir, or through the desired experiment-selections of the experiment-panel, without entering entry-values. The history of values for each of the variables of interest can be stored in the form of a matrix-like structure that is an array of arrays of values for each variable or parameter. The values of those arrays are transient values that can be used for statistical and sensitivity analysis within this system, or can be archived to a text file external to the system of this invention, or the values can be transferred to an external statistical package for further analysis. In addition, if the values of those arrays are desired in a permanent form within this system, this is accomplished by using the procedure provided to make the initial values of an array equal to the list of its current values, separated by commas.

[0235] The basal-density-proc has an important inference role since the value of the Basal-Density (or the Basal-Concentration) is used when the simulation is started as the initial value for the state variable of the bioPool, the Accumulation. The task of this procedure is to obtain a value for that attribute, if the user has not specifically entered one, from other data and information contained in other attributes of that bioReservoir or its bioPool. It also uses a set of lists encapsulated in the structure labeled BA (3317) upon the Simulation Panel to sort and list all the bioReservoirs to be included in the simulation, to indicate the source of those values used for the Basal-Density. The user can examine the results of that inference after the processing associated with the ACTIVATE button is completed, when the button becomes disabled and its appearance changes.

[0236] Selecting the BA button (3317) displays its subworkspace (3523) with a set of lists filled by the program following the criteria discussed below. Those lists can now be examined by selecting their icons to find out which of the bioReservoirs got the value from each of the alternative more or less precise sources. Various alternative versions of this procedure are made available, which are disabled and the one preferred is enabled by the modeler to better match the particular application. Alternatively, each of the alternative versions could be made specific for particular subclasses or groups of bioReservoirs. This procedure initially scans the normal-basal-density value of the bioReservoir and if that value is different from the default value of 9.99e-99 then the Basal-Density is set equal to its value, and the name of the bioReservoir is added to the NBD-BC list. If the subclass of bioReservoir is sol-mol-reservoir then the Basal-Concentration would be set equal to the value of the normal-basal-concentration if different from the default value. In the alternative implementation shown here, if the subclass of bioReservoir is sol-mol-reservoir this procedure scans the normal-basal-concentration value of the bioReservoir, if that value is different from the default value, 9.99e-99, then the Basal-Density is set equal to its value multiplied by the Avogadro's number, which states that each mol has 6.023e23 molecules, and the name of the bioReservoir is added to the NBC-BC list. The conversion is made when modeling complex mixtures of bound and soluble molecules. For example, for many biochemical systems where the molecules form large localized active complexes molecules are not homogeneously distributed, and therefore it is more meaningful to integrate in a bioProcess the quantities of both types of bioPools representing bound and soluble molecules using the number of molecules rather that their concentrations. If the normal-basal-density (or normal-basal-concentration) has the default value, then this procedure scans for the Scaling-Density attribute of the bioPool and if it has the default value then the bioReservoir is added to the "NO BC" list. Otherwise, this procedure scans for the

Scaled-Basal-Amount of the bioPool, and if its value is not the default value, then the Basal-Density is set equal to the value resulting from multiplying the Scaling-Density by the Scaled-Basal-Amount values. and the name of the bioReservoir is added to the SBA-BC list. Otherwise the Basal-Density is set equal to the value resulting from multiplying the Scaling-Density by a factor associated according to the symbolic value of the Abundance attribute of the bioReservoir, which value is set by the modeler or otherwise the default value is used, and the name of the bioReservoir is added to the PAB-BC list. The inference engine may proceeds the search for other sources of values, such as in an alternative method, where this procedure sets the value of the Basal-Density equal to a value proportional to the physiol-max-density (or physiol-max-concentration) of the bioReservoir if this value is different from default. If the user has not entered any of those values, then the program sets the value of the Basal-Density equal to the average of any known thresholding parameters of any bioReactant connected as an output of this bioReservoir, such as the Km, Ks, Ki or 1/(2*Kp). This alternative method emulates the following reasoning: the value of Basal-Density represents physiological conditions and it is assumed that physiological concentrations approach the mid-range functional concentrations in the most relevant bioEngine in which the bioReservoir participates, as represented by constants such as the Km for a substrate, the Ki for an inhibitor or the Ks for a ligand or complex-subunit. If those values are not known, a message is displayed requesting the user to assumed some values, based on comparisons with similar systems. Under normal in vitro assay conditions, an enzyme is generally present in limiting or catalytic amounts, in a range between 10e-3 nM and 10e2 nM, while a substrate is generally in the range of 10e3 nM to 10e7 nM. This value can be adjusted at a later time as more is learned about the system. If none of the alternative means are provided the system's default values are set.

[0237] Once the participants in the model have been activated and configured, the input values have been entered, and the model-definition has been configured, the simulation is started by selecting the START button (3322), which depending on the type of Simulation Panel will call one of the six procedures specific for each of the one-time Panels or for each of the periodic Panels, which read the input values entered by the user and perform different types of processing depending on the types of variables involved. The concentration-entry-panel is used to input the absolute concentration value entered by the user in the input-value edit-box, in units such as molar, which is added as a concentration or is converted and added as a density (depending of the alternative options available in this implementation). In a similar way the density-entry-panel is used to input an absolute density value, such as units/compartment, while the relative-entry-panel is used to input the scaled dimensionless value. Those input values are used the values of the corresponding attributes in each type of bioPool, such as the density-entry or the scaled entry at the time of entry value entered by the user, where these values are then integrated by the formula of the Accumulation of such bioPool at the next simulation cycle, and the values of such attributes are reset to 0.0 after one simulation period has elapsed since the time the input values were set. The set of periodic type of Panel follow initially the same type of processing, but the setting of the values of the density-entry or the scaled entry attributes of the bioPool occurs then repeatedly as many times as entered by the user in the entry frequency edit-box at times intervals as entered by the user in the time-interval edit-box, beginning at the simulation time entered by the user in the input-time edit-box. Each of those procedures-then call start-simulation-proc, which requests the simulator to run the model as previously defined in the model-definition.

[0238] During the simulation run the model-simulation-proc is called once every simulation cycle, which sets the values of the attributes current-simulation-time and current-simulation-time-increment of the entry-panel. The user can select the "T" button (3325) to display its subworkspace (3418), which contains a digital display of those values as well as the values of other performance parameters. This procedure also calls the compute-graph-transform-proc described below. By selecting the PAUSE button (3323) or the RESUME button (3324), which call their associated procedures, the user can pause and resume the simulation at will. By selecting the RESET button (3326) the reset-simulation-callback resets all the buttons on the Panel and calls the reset-proc which: a) first it scans the DBRL, UBRL, and DBPL lists and resets each of the bioReservoirs and bioProducts listed, including their status attributes, their icons and de-activating their subworkspaces; and b) then delete those lists. The DELETE button (3327) is disabled when the LISTS button is selected and it is not enabled until the reset is completed, preventing the user from deleting the panel before all the participants in the simulation that may have been modified are reset. The procedures are modified for the alternative implementation of the simulation, when copies are used instead of the original bioProcesses and bioReservoirs so that the values of the originals are not modified.

Digital and Graphic Displays of Dynamic Values

[0239] This system's graphic interface allows simulations to be followed in different ways, including: a) creating pathways on a workspace where dummy copies of the bioProcesses involved, or full copies of both the bioReservoirs and bioProcesses involved (depending on the alternative method of simulation used) of all the participating), are located in sequence and the corresponding connections between them are drawn. These structures allow to further interactively navigate through the system as usual, including back to the originals from which the copies were made; b) animation by color changes of the activated components; c) display of the current values of desired variables in digital form; d) dynamically created graphs of a set of time series of the values of desired variables: and e) dynamic charts of the values of a variable versus another variable. The current values of the variables of each bioReservoir or bioProcess are by default displayed in digital form (3403 and 3404). To dynamically display the time course of the values those variables for the time frame selected during a simulation graph-tracers are used, which subworkspaces with a graph and associated structures are not stored, but are rather created or deleted when the user selects one of such tracers. A graph-tracer is an object connected either to a bioPool (3420 and 3421) or a bioEngine (3426), used to control the dynamic creation and display at run-time of a graph which plots the values over time of key variables or parameters of the bioObject where they are contained, scaled or absolute versions for bioReservoirs. Selecting the "show-plot" option calls the graph-tracer-handler-proc which depending on the class of the graph-tracer from which it was invoked calls either the scaled-BR-graph-proc, the absolute-BR-graph-proc, or the scaled-BP-graph-proc, which first create a subworkspace (3422 or 3427) for that tracer (3421 or 3426) with all its contents by cloning the subworkspace of the corresponding prebuilt master structure, and then configure the labels of the graphs (3423 and 3428) and the ref-bioreservoir or ref-bioprocess attribute of each of the graph-tranf-vars (3424 or 3429) upon such new subworkspace to refer specifically to that bioReservoir (3419) or bioProcess (3425), respectively.

[0240] The design of graphs and associated structures deal with plotting multiple variable which values differ by orders of magnitude on graph structures that do not provide multiple axis scales. In the current implementation, the system automatically adjusts the scale to the current values of the time-series to be plotted, using a system of intermediary variables to dynamically transform those disparate values into others that fit a common scale. This set of new variables are instances of subclasses of the class transf-factor-var, a float-variable. Among the attributes of transf-factor-var are: "transform", which value is given by an instance of the class graph-transf-var; and Ref-variable, which makes reference to the particular attribute of a bioObject that provides the input value. The Label and the Transform value are displayed. The role of the graph-transf-var is to hold a transformed value of the current value of the parameter or variable referred to by its Ref-variable. The transformed value is suitable for plotting within the scale of the related graph. The role of the transf-factor-var is to hold the value of the factor necessary to obtain the desired transformation, as inferred by the compute-graph-transforms-proc only when a simulation-model is running. Each subclass of the class transf-factor-var corresponds to one of the bioObject's attributes to be transformed, which matches the value defined for the Label attribute of each subclass. The transf-factor-var provides by itself the order of magnitude of the variable it transforms. Selecting the "hide-s.graph" option of the bioReservoir, which appears only when the graph upon the subworkspace of the scaled-BR-graph-tracer exists, or deselecting the graph-tracer, or selecting the delete option directly from the subworkspace of the tracer deletes that subworkspace with its contents. The same applies for the other classes of graph tracers.

Scaled Computation Approach for the Variables of bioReservoirs and bioProcesses

[0241] The formulas of the parameters and variables that dynamically characterize a bioPool, depend on other numeric variables or parameters that are either attributes of the bioPool (FIG.13) or its bioReservoir (FIG.12), or that are attributes of the bioReactants (FIG.14) and bioProducts (FIG.15) that represent such bioReservoir in different bioProcesses. Table 1 lists a representative set of formulas, as examples of generic simulation formulas comprised in this invention to simulate a set of scaled-valued variables and Table 2 lists a set of generic simulation formulas specific for the Contribution of different subclasses of bioReactants that incorporate the values of scaled-valued variables or parameters. Table 5 lists a set of generic simulation formulas that are alternatives to the formulas in the set shown in Table 2, and which can be used in this invention to incorporate the values of absolute-valued variables or parameters into the Contribution of each bioReactant. These formulas for the Contribution do scale those absolute values using either a linear or sigmoid approach, depending the role in a bioProcess represented by each class of bioReactant. Therefore, since the output of the Contribution is dimensionless, whether the inputs are scaled (as in the case of using the formulas in Table 2) or absolute (as in the case of using the formulas in Table 5), it is now possible to integrate in a single bioProcess a mixture of bioReservoirs, where some of those bioReservoirs use only the scaled-valued variables while other bioReservoirs, called of mixed-type, use a mix of the scaled-valued variables, such as the Scaled-Amount, and absolute-valued variables, such as the Density. Those two values are interconvertible by using the value of the Scaling-Density, as computed by the formula for the Density in Table 5, so the user may be able to display the output of a simulation also as converted back into Density units in this case. The Contribution can be used in conjunction with either absolute-valued or scaled-valued variables, and therefore allows to integrate parts of the simulation where the absolute values of parameters and initial conditions are known, where other parts of the simulation where those values are unknown an have to be approximated, which is best done when using dimensionless values.

[0242] The Quantity of the bioPool is the only quantitative output value propagated from any component of a bioReservoir to any component of a bioProcess. That output quantity value of the bioPool is used to compute either: a) the value of the Velocity of each bioEngine connected to each bioReactant connected to such bioPool, which is the method normally used when using the absolute computation approach, to be discussed below; and b) the value of the Contribution of each bioReactant connected to such bioPool, which is an intermediary variable which is used to compute the value of the Velocity of the bioEngine to which that bioReactant is connected, in combination with the Contributions of all bioReactants connected to such bioEngine, this being the preferred method used when the scaled reasoning mode of simulation is preferred by the modeler or user. The formulas for the variables that give the value for the Contribution are specific for different classes of bioReactants, as more specifically defined in Tables 1 and 2. The formulas for the classes that represent reactants that bind to specific sites other reactants, such as substrate.r, ligand.r, inhibitor.r or ion.r, are designed as to provide a sigmoid scaling around their characteristic kinetic parameters, which implicitly provides a smoothed thresholding of the Quantity of the bioPool for that particular bioProcess, as represented by the bioReactant. When using the scaled-valued set of variables, the Scaled-Amount is used in conjunction with the corresponding scaled kinetic parameter, as the examples given in Table 5 show. The formulas for the Contribution for other classes of bioReactant, such as receptor.r or enzyme.r, are designed as to provide a linear scaling from none to their maximum observed value, represented as 100 if such absolute value is unknown. In contrast with the absolute mode of computation, where there may be great variability in the formulas needed to provide the value for the Velocity, depending on the class of bioEngine involved, in the scaled mode of computation the variability is introduced, to a much lesser extent in the simulation formulas that provide the value for the Contribution, while only one very generic simulation formula is required for the Velocity of all classes of bioEngines. As listed in Table 1, the very generic simulation formula for the Velocity included in the current implementation of this invention integrates various coefficients and a rate-constant to give the modeler or user the flexibility of by changing the values of those attributes of the bioEngine to modify the behavior of the system in different ways, without the need to write specific formulas for each situation, while the default values of those attributes do not affect the system.

[0243] The Velocity of each bioEngine is used to compute: a) the Production-Rate any bioProduct connected to such bioEngine; and b) the Consumption-Rate any bioReactant connected to such bioEngine, which may be 0.0 if such bioReactant represents a reactant that is not consumed in such process, or may have a value that may be withdrawn from a bioPool as an output (like any standard Consumption-Rate), but then retained for a period of time and then returned back to the bioPool as an input (like any standard Production-Rate will do), as specified by methods specific for the variables of such particular pair of bioReactant and bioPool, or for a group of such pair which are identifiable by some specific common attribute, or which are defined as separate subclasses of their respective classes. Several bioProcesses may contribute through their outputs to the Input-Rate of that bioPool, which values are then integrated when computing the value of the Accumulation of that bioPool. Users can also add (or remove) an absolute or scaled amount during a simulation to the selected bioPool, by having specific procedures setting the values of specific attributes of the bioPool, which are then integrated into the Accumulation by its simulation formula. The Basal-Quantity (which may be either a Basal-Concentration, a Basal-Density, or a Scaled-Basal-Amount) represents the initial conditions, and provides the initial-value for the integration of Accumulation, which is a state variable that integrates all other dependent variables. When using the scaled-valued set of variables, the value of the Scaled-Amount is the value of the Accumulation constrained to a range within 0.0 and 1.0. The values of the Concentration or Density are constrained to not be less that 0.0.

[0244] The implementation of the intermediary variable, the Contribution, is a novel and very important teaching of this invention, particularly in applications when the knowledge of the quantitative parameters and initial conditions of the system are incomplete, or when a more abstract and generic system is desired, such as when providing a Shell to be used in different and unpredicted ways. This implementation is relevant in several different ways. It allows to treat each of the bioReactant of a bioProcess as a generic self-contained unit, which allows to model also participants in a bioProcess which are known to interact but which exact roles in such process are unknown, where the velocity is generically computed by simply multiplying the contributions of any participant that the modeler may want to model. These generic units become even more generic when used in combination with the set of scaled-valued variables, which allow the user to enter values based on expert knowledge, or quickly perform what-if analysis. Another important result of using the Contribution is that it allows to perform quantitative simulations that integrate into the Velocity of any bioProcess values originated from bioReservoirs with both types of absolute-valued and scaled-valued variables. This means that those parts of the complex model where the absolute values of their kinetic parameters and initial conditions are known can be integrated with other parts of the complex model where relative values are used because the absolute values may be unknown. The abstraction provided by the Contribution also facilitates the use of bioProcesses as black-boxes, which are also be implemented in the current invention without using the Contribution. Black-boxes are used to represent any type of participant that is known to cause something to change somewhere down the line, even if it is known that other intermediaries are involved, but where the details or identity or even the existence of those intermediaries is unknown. The bioReactants in a black-box represent the entities (of bioPool(s)) that cause some effect on some other distant entities or bioProducts (the input to other bioPool(s)), and like in other generic bioProcesses, either the kinetic-coefficient (scaled or absolute) that modifies each bioReactant, or the rate-constant that modifies the overall process, representing the proportionality factor between the quantity of cause(s) and the quantity of effects, can be used.

Absolute Computation Approach for the Variables of bioReservoirs and bioProcesses

[0245] More standard ways of quantitative simulations using the known absolute values of the system parameters and initial conditions can be performed by using only the set of absolute-valued variables or parameters, and by using a different set of generic simulation formulas, such as those shown in Table 3. From those variables, the one that offers more variability in the formulas needed for each class of bioEngine is the Velocity, which is in contrast with the scaled mode of computation where only one very generic simulation formula is required for the Velocity of all classes of bioEngines. It is also possible to write very complex simulation formulas to encompass a variety of different situations, such as that in Table 4. The value of the Velocity of a bioEngine in this approach depends directly on the Quantities of the bioPools connected to each bioReactant connected to such bioEngine, which is the method normally used when the formulas that give the value of the Velocity are those that represent the equations that a biochemist will normally use to compute the velocity of a reaction, based on the quantities of the reactants and on some known kinetic parameters, such as the Michaelis constant (Km), inhibition constant (Ki) or the equilibrium dissociation constant, and which the preferred method used when the absolute reasoning mode of simulation is preferred by the modeler or user. The simulations formulas for the Velocity of absolute-valued enzymes, such as that listed in Table 3 (which is equivalent to the Briggs and Haldane approach where $V = n^* kp^*[E]^*[S]/([S] + Km * X)$, where $X = 1$ if no inhibitor.r is connected to the bioEngine, or $X = (1+ [I]/Ki)$ if an inhibitor.r is also connected to the bioEngine), assume the instantaneous or initial velocity approach, since increases or decreases in the Concentration of the substrate are updated at small time intervals in this invention.

Experiment Panel and Procedures

[0246] An Experiment Panel (3503) is a facility that allows to perform many of the tasks previously discussed, but with the additional capability of using multiple points of reference for the generation of lists, creation of pathways, or as the recipients of quantitative inputs for simulations. In contrast to the Simulation Panels, which are dynamically created and deleted when requested from the bioReservoir that serves as point of reference, a Library of Experiment Panels are created as permanent generic structures with different designs, accessible through the "Panels" head (3501) of the domain-menus associated with the General Mode and with the Simulation Mode previously discussed. Each Experiment Panel is the subworkspace of an instance of entry-panel created by cloning the Master-Experiment-Setup-Panel. As shown in FIG.35, selecting one of those options (3502) from the domain-menu (3501) displays the subworkspace (3503) of the named entry-panel, which corresponds to one of those predesigned Experiment Panels. The new components of such Panel are the structures (3504) organized in columns and rows which are instances of **class experiment-selection**. New designs are created by deleting of adding any number of such auxiliary structures, used to select the combination of bioReservoirs that are to serve as the points of reference in such experiments, and also to allow the user to enter the input values for quantitative simulations of such experiments.

[0247] Selecting one of the experiment-selections (3504) displays its table of attributes (3505) which allows to: a) enter a short name in its Label attribute displayed on top of the icon, b) select a bioReservoir, as entered in its Ref-bioreservoir slot, and c) enter whether Currently selected is true or false. The values of those attributes are sufficient to perform any of the qualitative tasks associated with this panel. If a quantitative simulation is desired, any of the values for the other optional attributes may be entered, such as those for an absolute-valued input or an scaled-valued input, time of input, time interval for periodic entries, and entry frequency, which have the same use as those previously discussed in relation to FIG.33. The configurations of any or all those experiment-selections can be stored for later use, and the combinations of different experiment-selections to be included in a particular experiment may be quickly changed at will by changing the value of the Currently selected, since only those with such value set to true, which get also highlighted with a different color, are included in the processing when any of the tasks is requested. This allows the user to have a set of experiment designs for those types of experiments most frequently used. The BIORESERVOIRS button (3506) provides a graphic tool to facilitate the task of the user in selecting the appropriate name to enter in the Ref-bioreservoir attribute of those experiment-selections, to be selected from among the large number of bioReservoirs in the Virtual Model. Selecting this button displays its subworkspace (3507) if one exists, or calls the BR-scroll-callback which creates one, containing an scroll-area listing in alphabetical order all the named bioReservoirs loaded on memory, and which in addition to providing the specific nomenclature used for desired bioReservoirs, it allows to directly access and verify the details of a selected bioReservoir.

[0248] Upon entering the choices of Ref-bioreservoir and setting the value true for the desired combination of experiment-selections, the user can proceed by selecting the PROCESSS button (3510), which calls the process-exper-callback which creates a set of lists and then scans the Panel to find all experiment-selections which Currently-selected is true and the Ref-bioreservoir has a value, which are inserted in the "Exper Selections" list (3512), and for each such experiment-selection found, it calls the same fill-br-downstream-lists-proc called from the Simulation Panel to fill the other two lists, one (3513) listing all downstream bioReservoirs of all those bioReservoirs named by the experiment-selections in the previous list, and the other (3514) listing all downstream bioProcesses of all those bioReservoirs named by the experiment-selections in the first list: and then finds all the bioModels that include all of those bioReservoirs and bioProcesses listed in those lists, and creates an scroll-area displaying those bioModels, to allow the user to select the desired scope of inclusion. Selecting the CLEAR button (3529) calls clear-exper-callback which deletes those lists and the scroll-area, to allow making different selections and repeating the PROCESS. The other buttons and structures on the Panel (3515, 3521-3530) have similar or equal functions as those described for the Simulation Panel, and therefore here only the differences will be briefly highlighted. Selecting the LIST button (3515) calls the list-exper-callback which creates a new set of lists: SDBRL (3516), SDBPL (3517), and UBRL (3518) lists. and finds the members of the previous sets of lists that are encapsulated in the selected bioModel. Selecting the PATHWAY button (3520) calls the draw-exper-pathway-callback which creates a navig-path-tracer (3531) together with its subworkspace by cloning the Master-Navig-Pathway and transfers it to the Panel, and then it scans the "Exper.Selections" list (3512) and for each bioReservoir listed calls the create-local-exper-BP-proc, while setting the value of the x-pos to a value close to the initial position when changing from one bioReservoir to the next, which performs a processing very similar to that described earlier for the create-local-UBP-proc for the upstream direction. Now, instead of once as before, that procedure is called as many times as bioReservoirs are in the "Exper.Selections" list.

Simulating Complex Models

[0249] The simulation can be continuous or time-constrained, and either way may be differentially applied to different parts of the complex model. To accomplish that, the simulation formulas can be complemented with rules that monitor those simulated variables to either set other values, particularly those that are further integrated into an state variable, or by controlling parts of the model. For example, a bioProcess or a bioReservoir can be programmed to be active only for certain time intervals during a simulation run, by means of rules or procedures that control the activation and de-activation of its subworkspace. Those time intervals may be either directly specified or dependent on the values of other parameters or variables or any of their combinations. Rules may be defined to monitor the simulated values of specific variables, such as the Quantity of a given bioPool or the Velocity of a given bioEngine, to be compared with a specified threshold, that when reached causes that rule to trigger some actions, such as setting the values of other parameters or variables, or activating and deactivating the subworkspace of one or more bioProcesses or bioReservoirs, or starting one or more procedures. Such rules may be: a) if-type rules with their Scan-interval attributes set to a specified time interval that preferably matches the time-interval used for the simulated variables, which means that such rules are invoked once every such interval; or b) whenever-type rules that are invoked whenever the specified variable receives a value and which trigger specified actions when such value fulfills any specified constraints; or c) initially-type rules, that may for example be hidden in the subworkspace of a bioEngine or bioPool, and which will be invoked once every time the subworkspace of their superior bioProcess or bioReservoir is activated, and which may trigger any of the actions described above which may apply to any bioProcess or bioReservoir in the complex model.

[0250] One of the several applications of this invention is that of a system for signal processing and signal integration. In such system, simulations can have three levels of control through: a) the interactions explicitly modeled in the architecture of the graphic design of this invention, namely the network of connected bioReservoirs and bioProcesses, which represent and determines the specific interactions of the different components of the complex model; b) the propagation through such network of the dynamic changes in the values of the variables of such components; c) through the signals transmitted by the smoothed thresholding implicit in the design of the Contribution variables of the bioReactants which determine the output values of the Velocity of the bioEngines, and d) through the rates of consumption of the bioReactants and the rates of production of the bioProducts, that determine new changes in the values of the Concentrations, now amplified to a larger number of bioPools, which initiate new waves of value propagations.

[0251] Tabular functions of one argument allow to deal with situations when the algebraic relationship between two bioVariables is not known but experimental data is available, which allows to build a table that relates a set of values that represents the magnitude of a cause with the set of values that represents the magnitude of the corresponding mechanistic effects or cellular responses, and straight-line interpolation is optional. These are important tools to biologist who frequently measure complex cellular responses to external factors.

5. Alternative Implementations and Variations

[0252] An equivalent system can be build with a less graphic interface by substituting each of the components represented in the current implementation as iconic bioObjects upon the subworkspace of bioReservoirs and bioProcesses, and representing them instead as attributes of the bioReservoirs and bioProcesses, respectively. The corresponding methods would need only minor modifications, such as for example, by substituting the current expression "the bioPool upon the subworkspace of bioReservoir X" with "the bioPool that is an attribute of bioReservoir X"; or substituting the current expression "the bioReactant-1 connected at port-1 of the bioEngine upon the subworkspace of bioProcess X" with "bioReactant-1 of bioReservoir X", where each of the specific bioReactants connected to each of the specific stubs of a specific type of bioEngine are now converted into specific attributes of the specific type of bioProcess, and where those attributes are not values but objects, and where those objects may have as attributes other objects. For example, the bioPosts now connected with each bioReactant and bioProduct can also be implemented as and attribute of such bioReactant or bioProduct, while the two sets of bioPosts connected to a bioPool can now become two sets of attributes objects of the bioReservoir, such as Input-1 or Output-3. By defining a specific attribute, such as Post for these new classes of objects, the specific distant connections can now be established by giving the same unique value to the Post attributes of one of the bioReactant attributes of a bioProcess and one of the Output attributes of a bioReservoir, or by giving the same unique value to the Post attributes of one of the bioProduct attributes of a bioProcess and one of the Input attributes of a bioReservoir. The result of this alternative implementation is that the different levels of encapsulation currently provided through the workspace hierarchy would then be provided by an attribute hierarchy, and the current representation of those components by their icons would be then limited to their representation by their tables of attributes, which are also part of the current implementation. Many of the tasks associated with the menus of the icons of those components or other auxiliary icons can be associated in the alternative implementation as menu options of the superior bioReservoirs and bioProcesses. as many of those are already currently implemented. The advantage of using the approach described in the currently preferred embodiment of this invention is that the more graphic interface is more intuitive for the user, and easier to use. The disadvantage is the additional memory required to store the additional graphical structures.

[0253] Another alternative is to replace the complex knowledge-structures bioReservoir and bioProcess by the components encapsulated in their respective subworkspaces. after some modifications. One of the several possible alternatives is to bypass the bioReservoir and the bioProcess structures altogether, and to encode the pertinent information here defined within the bioReservoir's table of attributes in the bioPool's table of attributes, and to encode the pertinent information here defined within the bioProcess's table of attributes in the bioEngine's table of attributes, and to construct the iconic components now contained in the subworkspaces of the bioReservoirs and bioProcesses directly on the desired workspace. The advantage of using the approach described in the currently preferred embodiment of this invention is that the bioReservoir and bioProcess structures encapsulate the details and simplifies the schematics, and makes easy the cloning and transfer of the preconstructed structures from one location to another, making the task of developing new models or modifying existing time much faster and keeping complex schematics simpler. The disadvantage is the additional memory required to store the additional graphical structures.

[0254] An alternative more compact implementation that would still allow developing multidimensional pathways, bypassing the need for bioReservoirs, is to directly connect all the bioReactant-post and bioproduct-post that in the current implementation are connected to the same bioPool, by giving them all the same name. In such implementation, the Ref-bioentity attribute, as currently defined for the class bioReservoir could instead be defined for the classes bioReactant and bioProduct, allowing in that way access to the bioEntity corresponding to an instance of bioReactant or bioProduct that would provide the physical description of such bioReactant or bioProduct, allowing integrating the description of structure with function. Other alternatives include storing the bioReservoir or equivalent structure within the subworkspace of the bioEntity; or storing the bioEntity within the subworkspace of the bioReservoir. However, in the current implementation the bioEntities are stored independently of the bioReservoirs because in many cases it may not be necessary to store one bioEntity for each bioReservoir, and instead prototypic bioEntities can be used that are shared by many bioReservoirs. In such cases, another alternative is storing all the bioReservoirs that in the current implementation would refer to the same bioEntity in the subworkspace of such bioEntity.

[0255] The concepts and methods subject of this invention can be applied with each of the object-oriented expert system shells from various vendors, after those capabilities not provided by that particular shell are additionally encoded. An equivalent of those capabilities of the selected shell that are required to apply the concepts and methods of this invention may also be custom built from scratch or by combining a set of off-the-shelf components. The methods here represented could in part be compiled using this Shell or other shells of the compiled type to improve performance at runtime by eliminating graphical interpretation of the iconic components. However, compilation does not facilitate frequent modifications of the design and information encapsulated in the iconic components, a feature which is of great importance when modeling biochemical systems, the knowledge of which is being continually updated and improved.

[0256] The same concepts described in this invention can be slightly modified by a person skilled in the art, to incorporate multiple inheritance capabilities, allowing the bioObjects to inherit attributes from two or more superior classes. Additional controls and code may be required in that case to resolve conflicts that may appear when more than one superior class.have the same attributes but receiving values from different sources, or through different methods.

[0257] Many of the symbolic variables or parameters could be also represented as logical variables or parameters, with or without fuzzy-beliefs, by simply changing the language of the attribute name and its values. To further control the behavior of any component of a simulation, different types of filters could be applied to, for example, the Density (or Concentration) of bioPools or to the Velocity of bioEngines. These filters can be of any of the types of filters used in signal processing control systems, such as low-pass or ideal high-pass filters or Gaussian filters. Such filters could be executed in either formulas or procedures, that could be generic for a whole class of variables, or more specific for certain instances or group of instances. The specific filters are more useful if fine control is required. The Density (or Concentration) of a bioPool is the result of multiple interactions and is represented as a differential equation in the present embodiment. Alternatively, it could be represented as a discrete-time linear system or a continuous-time linear system of equations, or whenever the Shell is able to handle vectors, as their equivalent matrix system. Difference or differential equations can be converted to state form, and the state variables of the system are them described as a state vector, the coefficients of the variables are described as a system matrix and the forcing terms are described as the forcing vector, following standard procedures.

7. Appendix

[0258]

TABLE 1

GENERIC-SIMULATION-FORMULAS for Scaled-Valued Variables
the SCALED-AMOUNT C of any BIOPOOL PO = (max (9.9e-9, the accumulation of PO))
state variable : d / dt (the ACCUMULATION C of any BIOPOOL PO) = the input-rate of PO - the output-rate of PO + the scaled-entry of PO - the decay-rate-factor of the bioreservoir superior to the workspace of PO * the scaled-amount of PO, with initial value the scaled-basal-amount of the bioreservoir superior to he workspace of PO
the INPUT-RATE of any BIOPOOL PO = the first of the following expressions that has a value ((max (0.0, the sum over each bioproduct P connected to PO of (the production-rate of P) + the first of the following expressions that has a value (the output-1 of the model-block M upon the subworkspace of the scaled-input-model-box connected to PO, 0.0))) , 9.9e-9)
the OUTPUT-RATE of any BIOPOOL PO = the first of the following expressions that has a value (max (0.0, the sum over each bioreactant R connected to PO of (the consumption-rate of R) + the first of the following expressions that has a value (the output-1 of the model-block M upon the subworkspace of the output-model-box connected to PO, 0.0)) , 9.9e-9)
the CONSUMPTION-RATE of any BIOREACTANT R = the first of the following expressions that has a value (max (0.0, the velocity of the bioengine connected to R / the stoichiometric-coeff of R), 9.9e-9)
the CONSUMPTION-RATE of any AMPLIFIER-BIOREACTANT R = 0.0
the PRODUCTION-RATE of any BIOPRODUCT P = the first of the following expressions that has a value (max (0.0, the velocity of the bioengine connected to P * the stoichiometric-coeff of P), 9.9e-9)
the VELOCITY of any BIOENGINE E = the first of the following expressions that has a value (max (0.0, the product over each bioreactant R connected to E of (the alpha-coeff of R * the contribution of R) * the rate-constant-sec of E * the tau-coeff of E + the bias of E), 9.9e-9)

TABLE 2

GENERIC-SIMULATION-FORMULAS FOR THE CONTRIBUTION
Based on Scaled-Valued Variables and Parameters

the CONTRIBUTION of any AMPLIFIER-BIOREACTANT R = the first of the following expressions that has a value (min (1 0, max (0 0, the effective-binding-sites of R * the scaled-amount of the biopool connected to R)), 0.5)

---

the CONTRIBUTION of any SUBSTRATE.R R = the first of the following expressions that has a value (min (1.0, max (0.0, the scaled-amount Q of the biopool PO connected to R / (the scaled-michaelis.k of R + Q))) , 0.5)

---

the CONTRIBUTION of any INHIBITOR.BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the scaled-amount Q of the biopool PO connected to R / (the scaled-inhibition k of R + Q))) , 0.5)

---

the CONTRIBUTION of any LEADING-BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the effective-binding-sites of R * the scaled-amount of the biopool connected to R)) ^ the stoichiometric-coeff of R, 0.5)

---

the CONTRIBUTION of any BINDING-BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the scaled-amount Q of the biopool PO connected to R / (the scaled equil.dissoc.k of R + Q))) ^ the stoichiometric-coeff of R , 0.5)

---

the CONTRIBUTION of any ION.R R = the first of the following expressions that has a value (min (1.0, max (0.0, the scaled-amount Q of the biopool PO connected to R / (the scaled-equilibrium.k of R + Q))) , 0.5)

---

the CONTRIBUTION of any SINGLE-BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the scaled-amount of the biopool connected to R)), 0.5)

TABLE 3

GENERIC SIMULATION FORMULAS FOR ABSOLUTE-VALUED VARIABLES
the CONCENTRATION of any SOLUBLE-MOL-POOL PO = (max (0.0, the accumulation of PO))
the DENSITY of any COMPLEXED-MOL-POOL PO = (max (0.0, the accumulation of PO))
state variable : d / dt (the ACCUMULATION of any SOL-MOL-POOL PO) = the input-rate of PO - the output-rate of PO + the concentration-entry of PO - the decay-rate-factor of the sol-mol-reservoir superior to the workspace of PO * the concentration of PO, with initial value the basal-concentration of PO
state variable : d / dt (the ACCUMULATION of any COMPLEXED-MOL-POOL PO) = the input-rate of PO - the output- rate of PO + the density-entry of PO - the decay-rate- factor of the bound-mol-reservoir superior to the workspace of PO * the density of PO, with initial value the basal-density of PO
the INPUT-RATE of any BIOPOOL PO = the first of the following expressions that has a value (max (0.0, the sum over each bioproduct P connected to PO of (the production-rate of P) + the first of the following expressions that has a value (the output-1 of the model-block M upon the subworkspace of the input-model-box connected to PO, 0.0)), 0.0)
the OUTPUT-RATE of any BIOPOOL PO = the first of the following expressions that has a value (max (0.0, the sum over each bioreactant R connected to PO of (the consumption-rate of R)+the first of the following expressions that has a value (the output-1 of the model-block M upon the subworkspace of the output-model-box connected to PO, 0.0)), 0.0)
the PRODUCTION-RATE of any BIOPRODUCT P = the first of the following expressions that has a value (max (0.0, the velocity of the bioengine connected to P * the stoichiometric-coeff of P), 0.0)
he CONSUMPTION-RATE of any BIOREACTANT R = the first of the following expressions that has a value (max (0.0, the velocity of the bioengine connected to R / the stoichiometric-coeff of R), 0.0)
the CONSUMPTION-RATE of any AMPLIFIER-BIOREACTANT R = 0.0
the VELOCITY of any BIOENGINE.E1.S1.I1 E = the first of the following expressions that has a value (the effective-binding-sites of the enzyme.r Z connected at the port-1 of E * the catalytic-rate-constant of Z * (the concentration of the sol-mol-pool connected to Z * the concentration SQ of the sol-mol-pool connected to the substrate.r S connected at the s1 of E / SQ + the michaelis-constant of S * (if there exists a comp-inhibitor.r I connected at the i1 of E then (1 + the concentration of the sol-mol-pool conneceted to I / the Inhibition-constant of I) else 1)), 1.0e-99)

TABLE 4

```
the VELOCITY of any BIOENGINE.E1.S1.I1 E = the first of the following expressions that has a value (the
effective-binding-sites of the enzyme.r Z connected at the port-1 of E * the catalytic-rate-constant of Z * (the
concentration of the soluble-mol-pool connected to Z * the concentration SQ of the soluble-mol-pool

 connected to the substrate.r S connected at the s1 of E /
 (if not (there exists an inhibitor.bioreactant connected at the i1 of E)
   then SQ + the michaelis-constant of S else
 (if there exists a comp-inhibitor.r CI connected at the i1 of E then SQ +
   the michaelis-constant of S * (1 + the concentration of the sol-mol-pool conneceted to I / the Inhibition-
 constant of I / the inhibition-constant of CI ) else
 (if there exists a noncomp-inhibitor.r NI connected at the i1 of E then
   SQ * (1 + the concentration of the sol-mol-pool conneceted to I / the Inhibition-constant of I  / the
 inhibition-constant of NI + the michaelis-constant  of S) * (1 + the concentration of the sol-mol-pool
 conneceted to I / the Inhibition-constant of I  / the Inhibition-constant of NI) else
 (if there exists an uncomp-inhibitor.r UI connected at the i1 of E then
   SQ * (1 + the concentration of the sol-mol-pool conneceted to I / the Inhibition-constant of I  / the
 inhibition-constant of UI + the michaelis-constant of  S) else
 (if there exists a mixed-inhibitor.r MI connected at the i1 of E then
    SQ * (1 + the concentration of the sol-mol-pool conneceted to I / the Inhibition-constant of I  / the
 inhibition-constant of MI + the michaelis-constant
    of S ) * (1 + the concentration of the sol-mol-pool conneceted to I / the Inhibition-constant of I / the
 inhibition-constant of MI ))))))), 0.0)
```

TABLE 5

GENERIC SIMULATION FORMULAS FOR MIXED-TYPE SIMULATIONS state variable : d / dt (the ACCUMULATION of any BIOPOOL PO) = the input-rate of PO - the output-rate of PO + the density-entry of PO - the decay-rate-factor of the bioreservoir superior to the workspace of PO * the density of PO, with initial value the basal-density of PO
the DENSITY of any BIOPOOL PO = the first of the following expressions that has a value (max (1.0e-9, (the scaled-amount of PO * the scaling-density of PO)), 9.9e-99)
the CONCENTRATION C of any BIOPOOL PO = the first of the following expressions that has a value (max 1.0e-9, (the density of PO / (6.023e23 * top-volume-po(PO)))) , 9.9e-99)
the CONTRIBUTION of any AMPLIFIER-BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the effective-binding-sites of R * the density of the biopool PO connected to R / the scaling-density of PO)), 0.5)
the CONTRIBUTION of any SUBSTRATE.R R = the first of the following expressions that has a value (min (1.0, max (0.0, the concentration Q of the biopool PO connected to R/ (the michaelis-constant of R + Q))) , 0.5)
the CONTRIBUTION of any INHIBITOR.BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the concentration Q of the biopool PO connected to R / (the inhibition-constant of R + Q))) , 0.5)
the CONTRIBUTION of any LEADING-BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the effective-binding-sites of R * the density of the biopool PO connected to R / the scaling-density of PO)) ^ the stoichiometric-coeff of R, 0.5)
the CONTRIBUTION of any BINDING-BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the density Q of the biopool PO connected to R / (the equilibrium-dissociation-constant of R + Q))) ^ the stoichiometric-coeff of R , 0.5)
the CONTRIBUTION of any ION.R R = the first of the following expressions that has a value (min (1.0, max (0.0, the concentration Q of the biopool PO connected to R/(the equilibrium-constant of R + Q))), 0.5)
the CONTRIBUTION of any SINGLE-BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the density of the biopool PO connected to R / the scaling-density of PO)), 0.5)

TABLE 6

GENERIC SIMULATION FORMULAS for Model Blocks the output-1 of any CONSTANT.F CF = the constant of CF
Tracing and breakpoints default the output-1 of any INCREASING.F = the current time * 1e-5
the output-1 of any PROPORTIONAL.F PF = the input-1 of PF * the gain of PF + the bias of PF
the output-1 of any SIGMOID.F SF =1.0/(1.0 + exp (- the gain of SF * (10 * the input-1 of S) - the bias of SF))
the output-1 of any EXP.C.PDF C = 1.0 - exp (- the x-val of E / the mean of E)
the output-1 of any EXP.I.PDF E = exp (- the x-val X of E / the mean M of E) / M
the output-1 of any UNIF.C.PDF UC = (if the x-val X of UC < the min-val MI of UC then MI else (if X > the max-val MA of UC then MA else ((X - MI) / (MA - MI))))
the output-1 of any UNIF.RN.F UR = random (the min-val of UR, the max-val of UR)
the output-1 of any NORMAL.RN.PDF NR = the mean of NR + ((the max-val of NR + the min-val of NR) / 6) * sqrt (-2.0 * ln (random (0.0, 1.0))) * sin (2.0 * pi () * random (0.0, 1.0)) *ln (random (0.0, 1.0))
the output-1 of any NORMAL.I.PDF NI = exp ((expt ((the x-val of NI - the mean of NI), 2)) / (2.0 * (the max-val MA of NI + the min-val MI of NI) / 6)) * (1.0 / (sqrt (2.0 * pi () * (MA + MI) / 6)))
the output-1 of any SIN.TIME.F ST = max (the min-val of ST, min (the max-val of ST, the bias of ST + the gain of ST * sin (the current time / the frequency of ST)))
the output-1 of any COS.TIME.F CT = max (the min-val of CT, min (the max-val of CT, cos (the current time /the frequency of CT) / the gain of CT))
the output-1 of any EXP.RN.PDF E = max (the min-val of E, min (the max-val of E, -ln (random (0.0, 1.0)) * the mean of E))

TABLE 7

GENERIC SIMULATION FORMULAS for Cell Cycle
state variable : d / dt (the number-of-cells of any g1-phase P) = the progression-rate of the cycle-path connected at the G0 of P + the progression-rate of the cycle-path connected at the Di of P - the progression-rate of the cycle-path connected at the S of P, with initial value 0.0
state variable : d / dt (the number-of-cells of any S-phase P) = the progression-rate of the cycle-path connected at the G1 of P - the progression-rate of the c-cycle-path connected at the G2 of P, with initial value 0.0
state variable : d / dt (the number-of-cells of any G2-phase P) = the progression-rate of the cycle-path connected at the S of P - the progression-rate of the c-cycle-path connected at the M of P - the progression-rate of the cycle-path connected at the Ap of P, with initial value 0.0
state variable : d / dt (the number-of-cells of any M-phase P) = the progression-rate of the cycle-path connected at the G2 of P - the progression-rate of the cycle-path connected at the G0 of P - the progression-rate of the cycle-path connected at the G1 of P, with initial value 0.0
state variable : d / dt (the number-of-cells of any differ-stage P) = the progression-rate of the cycle-path connected at the G1 of P - the progression-rate of the cycle-path connected at the NG of P, w. initial value 0.0
state variable : d / dt (the number-of-cells of any apoptosis P) = the progression-rate of the cycle-path connected at the G2 of P, with initial value 0.0
state variable : d / dt (the number-of-cells of any cell-phase C) = the sum over each cycle- path Pi connected at an input of C of (the progression-rate of Pi) - the sum over each cycle-path Po connected at an output of C of (the progression-rate of Po) , with initial value 0.0
the progression-rate of any c-cycle-path P = the rate-constant of P * the tau-coeff of P * the number-of-cells of the cell-phase connected at the input of P

TABLE 8

Class name Superior class Attributes specific to class Class restrictions	inference-block object label is ""; description is ""; outcome is given by an outcome-par; beta-coeff is given by a beta-coeff-par when in simulation, navigation, or explorer mode: menu choices exclude additionally: move, name, change-size, color, describe, create-subworkspace, rotate-reflect, clone, delete; attributes visible exclude additionally: notes, names, user-restrictions
Class name Superior class	deactivation-block inference-block

TABLE 8 (continued)

Attributes specific to class	none
Class name Superior class Attributes specific to class	time-deactivation-block deactivation-block time-interval is given by a deactivation-interval-par; time-threshold is 9.9e99
Class name Superior class Attributes specific to class	event-deactivation-block deactivation-block none
Class name Superior class Attributes specific to class	activation-block inference-block none
Class name Superior class Attributes specific to class	time-activation-block activation-block time-interval is given by an activation-interval-par; time-threshold is 9.9e99
Class name Superior class Attributes specific to class	checkpoint-to-floor-block activation-block checkpoint is given by a checkpoint-var; threshold is 9.9e99
Class name Superior class Attributes specific to class	checkpoint-to-ceiling-block activation-block checkpoint is given by a checkpoint-var threshold is 9.9e99

TABLE 9

INFERENCE BLOCK RULES

```
for any event-deactivation-block B that is an attribute of any time-compartment CC
  for any time-compartment NC connected at the next-1 of CC
  whenever the status S of NC receives a value and when S is activated then
     deactivate the subworkspace of CC and conclude that the status of CC is deactivated
```

---

whenever the time-interval I of any time-activation-block B receives a value and when the beta-coeff of B * I >= the time-threshold of B then conclude that the outcome of B is above-threshold

---

whenever the time-interval I of any time-deactivation-block B receives a value and when the beta-coeff of B * I >= the time-threshold of B then conclude that the outcome of B is above-threshold

---

whenever the checkpoint C of any checkpoint-to-floor-block B receives a value and when the beta-coeff of B * C <= the threshold of B then conclude that the outcome of B is below-threshold

---

whenever the checkpoint C of any checkpoint-to-ceiling-block B receives a value and when the beta-coeff of B * C >= the threshold of B then conclude that the outcome of B is above-threshold

---

TABLE 10

Cell Cycle Rules

for any g0-phase P
whenever the outcome O of the next-activation-block of P receives a value and when O is not within-threshold then activate the subworkspace of the g1-phase G connected at the g1 of P and conclude that the status of G is activated

---

for any g1-phase P
whenever the outcome O of the differentiation-activation-block of P receives a value and when the status of the s-phase connected at the s of P is deactivated and O is not within-threshold then activate the subworkspace of the differ-stage N connected at the di of P and conclude that the status of N is activated

---

for any g1-phase P
whenever the outcome O of the next-activation-block of P receives a value and when the status of the differ-stage connected at the di of P is deactivated and O is not within-threshold then activate the subworkspace of the s-phase N connected at the s of P and conclude that the status of N is activated

---

for any s-phase P
whenever the outcome O of the next-activation-block of P receives a value and when O is not within-threshold then activate the subworkspace of the g2-phase G connected at the g2 of P and conclude that the status of G is activated

---

for any g2-phase P
whenever the outcome O of the apoptosis-activation-block of P receives a value and when the status of the m-phase connected at the m of P is deactivated and O is not within-threshold thenactivate the subworkspace of the apoptosis N connected at the ap of P and conclude that the status of N is activated

---

for any g2-phase P
whenever the outcome O of the next-activation-block of P receives a value and when the status of the apoptosis connected at the ap of P is deactivated and O is not within-threshold then activate the subworkspace of the M-phase N connected at the m of P and conclude that the status of N is activated

---

for any M-phase P
whenever the outcome O of the next-activation-block of P receives a value and when O is not within-threshold then conclude that the generation-number N of the g0-phase G connected at the g0 of P = N + 1 and conclude that the number-of-cells C of G = 2 * C and conclude that the mass-per-cell of G = the mass-per-cell of P / 2 and deactivate the subworkspace of P and conclude that the status of P is deactivated

---

for any M-phase P
whenever the outcome O of the g1-activation-block of P receives a value and when O is not within-

threshold then conclude that the generation-number N of the g0-phase G connected at the g0 of P = N + 1 and conclude that the number-of-cells C of G = 2 * C and conclude that the mass-per-cell of G = the mass-per-cell of P / 2 and activate the subworkspace of the g1-phase G connected at the g1 of P and conclude that the status of G is activated and deactivate the subworkspace of P and conclude that the status of P is deactivated

---

```
for any differ-stage P
  whenever the outcome O of the next-activation-block of P receives a value and when O is not within-
threshold then activate the subworkspace of the g0-phase G0 connected at the ng of P and conclude that the
status of G0 is activated and activate the subworkspace of the g1-phase G1 connected at the ng of P and
conclude that the status of G1 is activated
```

TABLE 11

Chaining Rules

Options for all whenever rules:	not invocable via backward chaining, not invocable via forward chaining, may cause data seeking, may cause forward chaining

---

```
for any bioprocess BP
    for any bioprocess BP1 that is an-upstream-bioprocess-of BP
        for any bioprocess BP2 that is an-upstream-bioprocess-of BP
whenever BP1 ceases to be an-upstream-bioprocess-of BP and when not (there exists a bioprocess BP3
such that (BP2 is an-upstream-bioprocess-of BP3 and BP3 is an-upstream-bioprocess-of BP)) then
conclude that BP2 is not A-DOWNSTREAM-BIOPROCESS-OF BP
```

```
for any bioprocess BP
    for any bioprocess BP1 that is an-upstream-bioprocess-of BP
whenever any bioprocess BP2 becomes an-upstream-bioprocess-of BP1 and when BP2 is not
an-upstream-bioprocess-of BP then conclude that BP2 is AN-UPSTREAM-BIOPROCESS-OF BP
```

```
for any bioprocess BP
    for any bioprocess BP1 that is a-downstream-bioprocess-of BP
        for any bioprocess BP2 that is a-downstream-bioprocess-of BP
whenever BP1 ceases to bea-downstream-bioprocess-of BP and when not (there exists a bioprocess BP3
such that (BP2 is a-downstream-bioprocess-of BP3 and BP3 is a-downstream-bioprocess-of BP)) then
conclude that BP2 is not A-DOWNSTREAM-BIOPROCESS-OF BP
```

```
for any bioprocess BP
    for any bioprocess BP1 that is a-downstream-bioprocess-of BP
whenever any bioprocess BP2 becomes a-downstream-bioprocess-of BP1 and when BP2 is not
a-downstream-bioprocess-of BP then conclude that BP2 is A-DOWNSTREAM-BIOPROCESS-OF BP
```

Claims

1. A computer-system for generating virtual models of complex-systems comprising processor means, memory, means, storage means, input means, output means, display means, and program means, characterized by:

   - one or more databases stored in said storage means comprising instances of process (502) and reservoir (501) building-blocks representing the components of one or more virtual model of said complex-systems, each of said instances comprising any number of attributes, which value types include but are not limited to character string, integer or real numbers, logical values, fuzzy values, or instances of variables, parameters, lists, arrays, images, or any other data structure, or pointers to any of said instances of building-blocks, external files, Uniform Resource Locators (URLs), database records, or any other data structure, in said computer system or in a network accessible by said computer system, wherein:

     - each of said instances of process building-blocks (502) represents one of the many processes of said complex-system and comprises at least one input (503) or one output (505) representing at least one inflow or one outflow of said process, wherein said process building-blocks may be of different types representing different types of processes;
     - each of said instances of reservoir building-blocks (501) represents a pool of any number of units of a single type or state of entities available to participate in one or more of said processes and comprises at least one input (506) or one output (507) representing at least one inflow or outflow of said pool, wherein each of said pools of entities do not need to be physically separated, in a reservoir or otherwise, from other pools of entities in said complex-system;
     - at least one input (503) or one output (505) of each of said instances of process building-blocks is linked to an output (507) or an input (506) of said instances of reservoir building-blocks, respectively, said links representing that said inflows or outflows from said processes are outflows or inflows of said pools of entities, respectively wherein each of said linked instances of reservoir building-blocks (501) has any number of input links (506) and/or output links (507) to any number of said linked instances of process building-blocks (502), and each of said linked instances of process building-blocks has any number of input links (507) and / or output links (506) to any number of said linked instances of reservoir building-blocks;
     - said program means having access to said stored database(s) comprising means to integrate said alternating instances of linked reservoir and process building-blocks as the nodes of one or more multidimensional networks (Fig.5) of pathways, each of said instances being comprised in one or more of said pathways, with each of said instances that have multiple input links being nodes where different pathways merge, and each of said instances that have multiple output links being nodes where different pathways branch, wherein information and data comprised in said building-blocks about the components of said of complex-system becomes integrated and related to each other through said series of links.

2. A computer-system as claimed in Claim 1 further enabling quantitative simulations of said virtual models in said computer-system, characterized by:

   - mathematical models characterizing the quantitative behavior of said complex-systems are decomposed into networks of quantitative structures distributed among said instances of building-blocks (Figs 12 through 15) comprising state variables, dependent variables, and parameters, representing characteristics of the components they represent. wherein the values of said parameters and the initial values of said state-variables are measured, calculated, or estimated from the components represented or are typical default values;
   - said program means comprise simulation means based on laws characteristic of said complex-systems for dynamically computing the current values of each of said variables over time by incorporating the value(s) of certain of said parameter(s) and/or variable(s) comprised in either the same instance of building-block of said variable and/or the instances of building-blocks linked said instance.

3. A computer-system as claimed in Claim 2 further enabling mixed-type simulations of said virtual models using quantitative or semi-quantitative data, or heuristic information, for different parts of said complex-systems wherein:

   - said instances of building-blocks further comprise a second set of their corresponding quantitative variables and/or parameters, one set having absolute values and the other set having scaled values,
   - said program means further comprise simulation means and/or inference means for dynamically computing over time and integrating said absolute-valued variables and parameters of certain parts of said virtual models

and said scaled-valued variables and parameters of other parts of said virtual models.

4. A computer-system as claimed in Claim 1 wherein said virtual models are compartmentalized in space, further comprising instances of location-compartment building-blocks of one or more types (2406, 2408) representing discrete physical or conceptual space compartments characteristic of said complex-systems, used to organize said instances of process (2410) and reservoir (2414) building-blocks of said virtual models into a hierarchy of one or more layers (2405, 2407, 2409) of said location-compartment building-blocks.

5. A computer-system as claimed in Claim 1 wherein said virtual models are compartmentalized in time intervals, further comprising instances of time-compartment building-blocks of one or more types (2421, 2423) representing discrete physical or conceptual time intervals characteristic of said complex-systems, used to organize said instances of process (2410) and reservoir (2414) building-blocks of said virtual models into a hierarchy of one or more layers (2402, 2422) of said time-compartment building-blocks.

6. A computer-system as claimed in Claim 1 further comprising one or more instances of entity building-blocks (18014), each instance representing the compositional description (1807) of a representative entity of a pool represented by an instance of said reservoir building-blocks, said instances of entity building-blocks being accessible from the corresponding instances of reservoir building-blocks (1801).

7. A computer-system as claimed in Claim 1, further comprising program means for dynamically generating interactive visual displays (3110, 3118, 36) of said networks of multidimensional pathways comprissing interconnected visual representations of instances of at least one of said types of building-blocks, wherein:

- the composition of the pathways in said displays is determined by said program means to represent either said complex-systems or any of their subsystems that meet user-selected criteria, such as: selecting any one or more of said instances of building-blocks as starting node(s) and selecting a direction upstream and/or downstream from said starting node (s); and
- the connected (3114, 3115) visual representations in said displays are either pointers to said stored linked instances of building-blocks or representations of clones of said stored linked instances of building-blocks.

8. A computer-system as claimed in Claim 1, further comprising program means for performing queries (Fig.30) based on said upstream or downstream successive links from any of said instances of building-blocks selected as reference.

9. A computer-system as claimed in Claim 1 wherein said process building-blocks (843, 837) are composite, each comprising a subworkspace (809, 836) with a set of components that can be displayed to provide visual access to said components comprising:

- one or more instances of reactant building-blocks (835), representing the different inflows to the process represented by said instance of process building-block, each linked (834) to an output of an instance of reservoir building-block (801), different types of reactant building-blocks (820, 825, 830, 835, 838, 839) representing different roles that said inflows play as participants in said processes;
- one or more instances of product building-blocks (808) representing the outflow(s) from the process represented by said process building-block, each linked to an input of a reservoir building-block (801).

10. A computer-system as claimed in Claim 9 wherein:

- said reservoir building-blocks (801) are composite, each comprising a subworkspace (804) with a set of components; linked to components of process building-blocks, that can be displayed to provide visual access to said components and to navigate throughout said virtual models, said components comprising:
  - one or more instances of input building-blocks (806), each representing one of said inflow(s) to said pool of entities, each linked to (807) one of said instances of product building-blocks (808); and
  - one or more instances of output building-blocks (833), each representing one of said outflow(s) from said pool of entities, each linked to (834) one of said instances of reactant building-blocks (835);
- navigation means associated with said linked instances of reactant product, input and output building-blocks allow to display the components of any instances of process or reservoir building-blocks upstream and/or

downstream of any selected instance.

11. A method for using virtual models (Fig.5) of complex-systems for designing monitoring and/or control systems in a computer-system (112) comprising processor means, memory means, storage means, input means, output means, display means, program means (114), and a stored database of modular building-blocks representing the components of one or more complex-systems, the method being characterized by:

- loading said database (116) into said memory means, said database comprising any number of instances of process (502) and reservoir (501) building-blocks, wherein:
    - each of said instances of process building-blocks (502) represents one the processes of said complex-system and comprises at least one input (503) or one output (505) representing at least one inflow or one outflow of said process, different types of said process building-blocks representing different types of said processes
    - each of said instances of reservoir building-blocks (501) represents a pool of any number of units of a single type or state of entities that is available to participate in one or more of said processes (502), and comprises at least one input (506) or one output (507) representing at least one inflow or outflow of said pool. each of said pools of entities not having to be physically separated in a reservoir or otherwise from other pools of entities in said complex-system;
    - any of said input(s) (503) or output(s) (505) of any of said instances of process building-blocks is linked to an output (507) or input (506), respectively, of an instance of said reservoir building-blocks (501); and any of said input(s) (506) or output(s) (507) of any of said instances of reservoir building-blocks are linked to any number of output(s) or input(s), respectively, of instances of process building-blocks;
    - a number of said alternating instances of said reservoir and process building-blocks linked in series are the nodes that define a pathway;
- selecting by said program means the set of instances of building-blocks stored in said database(s) that are the nodes of the pathways to be included in the virtual model of the subsystem to be used for monitoring and/or control, given user selections such as any one or more of said building-blocks as starting node(s) and a direction upstream and/or downstream from said starting node(s); and
- dynamically integrating by said program means said set of selected instances into a multidimensional network of pathways (Fig.5), each of said instances being comprised in one or more of said pathways, with each of said instances that has multiple input links or multiple output links being nodes where different pathways merge or branch, respectively.

12. A method as claimed in Claim 11 further enabling the generation of visual displays (Figs. 31, 36) of said networks of multidimensional pathways, further comprising the steps of:

- selecting by said program means the set of instances of user-selected type of said building-blocks of said virtual model(s) or any of its subsystems that meets other user-selected criteria, such as: selecting any one or more of said instances of building-blocks as starting node(s) and selecting a direction upstream and/or downstream from said starting node, to be included in said visual displays;
- creating by said program means visual representations (3114) that are pointers to, or alternatively visual representations of clones of, said selected stored instances of building-blocks;
- dynamically generating a layout (3110, 3118, Fig.36) on said display means with said visual representations connected as the nodes of a network of one or more pathways.

13. A method as claimed in Claim 11 for designing manipulation strategies to control the operation of complex bio-chemical-systems in applications comprising treatment of disease, improvement of livestock and food crops, and improvement of the environment among others, the method being further characterized by:

- searching for and unveiling by said program means any number of instances of said building-blocks of said virtual models that represent potential targets for manipulation to achieve any desired goal(s), such as affecting the expression of any desired gene(s), said set of instances being comprised in any pathway(s) upstream from the instance(s) of reservoir building-blocks representing the pool of entities involved in the accomplishment of said goal(s), such as said gene(s). as selected by a user.

14. A method as claimed in Claim 11 for designing manipulation strategies to control the operation of complex biochemical-systems by candidate external agent(s), real or designed, in applications comprising treatment of disease, improvement of livestock and food crops, and improvement of the environment among others, the method being further characterized by:

- expanding said database to comprise one or more instances of reservoir building-blocks representing pool(s) of any number of units of said external agent(s) and one or more instances of process building-blocks representing the interaction(s) of said external agent(s) with their target(s) within said complex biochemical-system (s);
- integrating by said program means said new instances of reservoir and process building-blocks representing said external agent(s) and their interaction(s) into said network of pathways;
- searching for and unveiling by said program means the set of instances of said building-blocks of said virtual models that could be affected by manipulating any one or any combination of said pools of external agent(s), said set of instances being comprised in any pathway(s) downstream from said instance(s) of reservoir building-blocks representing said external agent(s), as selected by a user.

15. A method as claimed in Claim 11 further enabling the simulation of the quantitative behavior of said virtual models in said computer-system, wherein each of said instances of building-blocks further comprises one or more quantitative variables parameters (Figs 12 through 15) that determine the quantitative behavior of the components of said complex-system(s) represented by said instances, and wherein said program means further comprises simulation means, the method being further characterized by:

- using said simulation means to dynamically simulate, in simulated-time or real-time, said virtual model(s) or their subsystems by computing over time the current values of each of said variables while incorporating the value(s) of certain of said parameter(s) and/or variable(s) comprised in either the instance of building-block containing said variable and/or the instances of building-blocks linked to said instance.

16. A method as claimed in Claim 15 further enabling mixed-type simulations of said virtual models using quantitative or semi-quantitative data, or heuristic information, for different parts of said complex-systems, wherein said instances of building-blocks further comprise a second set of their corresponding quantitative variables and/or parameters, one set having absolute values and the other set having scaled values the method being further characterized by:

- using said simulation means for dynamically computing over time and integrating during a simulation run said absolute-valued variables and parameters of certain parts of said virtual models and said scaled-valued variables and parameters of the other parts of said virtual models.

17. A method as claimed in Claim 15 further enabling the simulation of the quantitative behavior of complex biochemical-systems to monitor and evaluate the effects of manipulation strategies in applications comprising treatment of disease, improvement of livestock and food crops, and improvement of the environment among others, the method being further characterized by:

- selecting by said program means the set of instances of building-blocks stored in said database to be included in the virtual model of the subsystem to be simulated to evaluate the achievement of any desired goal(s), such as affecting the expression of any desired gene(s), said set of instances being comprised in any pathway(s) upstream from the instance(s) of reservoir building-blocks representing the pool of entities involved in the achievement of said goal(s), such as said gene(s), as selected by a user.

18. A method as claimed in Claim 15 further enabling the simulation of the quantitative behavior of complex biochemical-systems to monitor and evaluate the effects of external agent(s), real or designed, in applications comprising treatment of disease, improvement of livestock and food crops, and improvement of the environment among others, the method being further characterized by:

- expanding said database to comprise one or more instances of reservoir building-blocks representing pool(s) of any number of units of said external agent(s) and one or more instances of process building-blocks representing the interaction(s) of said external agent(s) with their target(s) within said complex biochemical-system (s);
- integrating by said program means of said instances of reservoir and process building-blocks representing said external agent(s) and their interaction(s) into said network of pathways; and

- selecting by said program means the set of instances of building-blocks stored in said database to be included in the virtual model of the subsystem to be simulated to evaluate the effects of manipulating any one or any combination of said external agent(s), said set of instances being comprised in any pathway(s) downstream from said instance(s) of reservoir building-blocks representing said external agent(s). as selected by a user.

19. A method as claimed in Claim 15 further enabling the synchronization of real-time simulations of said virtual models (116) in said computer-system with the real operation of said complex-system (102), by integrating (118) on-line monitored variable values (122) of any component(s) of said complex-system into said simulation, further comprising monitoring means (106, 108) linked (120) to said input means, the method being further characterized by:

- setting up and using said monitoring means to monitor (106) one or more variable value(s) characteristic of said processes or pools of entities of said complex-system(s) represented by any of said variable(s) (118) comprised in any of said instances of reservoir (501) or process (502) building-blocks of said virtual model;
- defining and using one or more monitoring interfaces (120) in said computer-system between said monitoring means (106, 108) and said program means (114) to relate and pass said monitored values (122) to their corresponding variable(s) (118) comprised in said instances of building-blocks;
- setting each of said variable(s) (118) representing said monitored value(s) to dynamically obtain during a simulation its values over time from said corresponding measured values, instead of said values being simulated.

20. A method as claimed in Claim 15 further enabling the control of a processing-system comprising one of said real complex-systems based on a real-time simulation of its virtual model (116), further comprising controller means (138) linked to said processing-system (102) and said output means (130), the method being further characterized by:

- setting up and using said controller means (138) to regulate the operation of desired components (140) of said processing system;
- defining and using controlling code to generate control signals to be passed (132) to said controller means (138), said signals triggered during a simulation when the simulated value(s) of certain one or more of said variables (128) comprised in said instances of building-blocks of said virtual model reach certain threshold value(s) or are within certain ranges;
- defining and using one or more controller interfaces (130) -in said computer-system between said program means and said controller means to pass said control signals to said controller means.

21. A method for building virtual models (Fig.5) of complex-systems in a computer-system comprising processor means, memory means, storage means, input means, output means, display means, program means, and a library (2001, 1901) of various modular prototypes of building-blocks with associated methods stored in said computer-system representing the various types of components of said complex-systems, the method being characterized by:

- creating by said program means instances of any of said prototypes selected by a user from said library to represent the various components of any of said complex-systems, wherein said library comprises:

    - one or more prototypes of reservoir building-blocks (1201), which instances are used to represent each a pool of any number of units of a single type or state of entities, comprising each at least one input or one output representing one inflow or outflow of said pool, wherein each of said pools of entities do not need to be physically separated, in a reservoir or otherwise, from other pools of entities in said complex-system; and
    - different prototypes of process building-blocks (2001), each representing one of the different types of processes characteristic of said complex-systems, each comprising at least one input and one output representing one inflow and one outflow of said process;

- establishing by said program means links between output-input pairs selected by a user, said output-input pairs comprising: an output of any of said instances of reservoir or process building-blocks and an input of any of said instances of process or reservoir building-blocks. respectively, representing that an outflow from the first is an inflow source to the second instance of said building-blocks;
- storing in said computer-system said instances of said prototypes and said links; and
- dynamically integrating by said program means said linked building-blocks into networks of multidimensional pathways (Fig.5) which nodes are said alternating linked instances of reservoir (501) and process (502) build-

ing-blocks that integrate and relate information and data about the components represented by said linked instances, wherein each of said instances of reservoir building-blocks has input links from and/or output links to any number of said instances of process building-blocks, and wherein each of said instances of process building-blocks has input links from and/or output links to any number of said reservoir building-blocks, each of said instances being comprised in one or more of said pathways, with each of said instances that have multiple input links being nodes where different pathways merge, and each of said instances that have multiple output links being nodes where different pathways branch.

22. A method as claimed in Claim 21 wherein each of said prototypes of building-blocks further comprises one or more quantitative state or dependent variables, or parameters (Figs 12 through 15) to enable quantitative simulations of said virtual models, the method being further characterized by:

- defining simulation means in said computer-system to compute over time the current values of each of said variables, by incorporating the value(s) of certain of said parameter(s) and/or variable(s) comprised in either the instance of building-block of said variable and/or the instances of building-blocks linked to said instance;
- setting by said program means the default values of said parameters of said instances of building-blocks to certain values defined in said program means for each of said prototypes based on typical normal values characteristic of the types of components of said complex-systems represented by said prototypes;
- using said input means to override said default values with the measured, calculated, or estimated values of any of said parameters of any of said instances for which said specific values are available or desired; and
- setting by said program means the initial values for any of said state variables based on the availability of certain other specific data or information stored in said instances of building-blocks, as defined in said program, or in their absence on default values defined in said program for each of said prototypes based on typical initial values characteristic of the types of components of said complex-systems represented by said prototypes.

23. A method as claimed in Claim 22 wherein said prototypes of building-blocks further comprise a second set of their corresponding quantitative state or dependent variables, and/or parameters, one set having absolute values and the other set having scaled values, to enable dynamic mixed-type simulations of said complex-systems with data of different degrees of precision available for different parts of said models, the method being further characterized by:

- defining further simulation means for dynamically integrating said absolute-valued variables and parameters of certain parts of said virtual models and said scaled-valued variables and parameters of other parts of said virtual models, using scaling functions specifically defined for the different types of building-blocks.

24. A method as claimed in Claim 21 further enabling building virtual models compartmentalized in space, wherein said library further comprises any number of prototypes of location-compartment building-blocks (2405, 2407) representing discrete physical or conceptual spaces characteristic of said complex-systems, the method being further characterized by:

- creating by said program means instances of said prototype(s) of location-compartment building-blocks selected by a user from said library, to build a hierarchy of any number of layers of said instances to represent the space hierarchy characteristic of the complex-system being modeled;
- organizing said instances of process (2410) and reservoir (2414) building-blocks of said virtual models to be comprised within appropriate lower-level instances (2408) of said location-compartment building-blocks in said hierarchy; and
- by said program means; containment relations between any of said instances of location-compartment building-blocks and other instances of location-compartment, process, and reservoir building-blocks that fulfill one or more criteria based on said hierarchical organization.

25. A method as claimed in Claim 21 further enabling building virtual models compartmentalized in time intervals, wherein said library further comprises any number of prototypes of time-compartment building-blocks (2404, 2421, 2423) representing discrete physical or conceptual time intervals characteristic of said complex-systems, the method being further characterized by:

- creating by said program means, instances of said prototype(s) of time-compartment building-blocks selected by a user from said library, to build a hierarchy of any number of layers (2402, 2422) of said instances to represent the time interval hierarchy characteristic of the complex-system being modeled;

- organizing said instances of process (2410) and reservoir (2414) building-blocks of said virtual models to be comprised within appropriate lower-level instances of said time-compartment building-blocks in said hierarchy; and
- establishing, by said program means: containment relations between any of said instances of time-compartment building-blocks and other instances of time-compartment, process, and reservoir building-blocks that fulfill one or more criteria based on said hierarchical organization.

26. A method as claimed in Claim 21 wherein said library further comprises any number of prototypes of entity building-blocks (Fig.18) representing types of entities characteristic of said complex-systems or types of components of said entities, the method being further characterized by:

- creating by said program means instances of said prototype(s) of entity building-blocks selected by a user from said library;
- organizing into each instance of top-level entity building-blocks (1804), each representing the compositional description of a typical entity, of any of said instances of reservoir building-blocks (1801), a hierarchy of any number of layers (1807, 1809, 1914) of said instances of entity, building-blocks representing its modular components;
- making by said program means said top-level instances accessible (1803) from the corresponding instances of reservoir building-blocks (1801) in which a pointer to any of said top-level instances has been entered.

27. A method as claimed in Claim 21 wherein said prototypes of process building-blocks (843, 837) are composite, comprising each a subworkspace (809, 836) with a set of components that can be displayed to provide visual access to said components, the method being further characterized by:

- said components comprise: one or more instances of reactant building-blocks (835) to represent the different inflows characteristic of the type of process represented by each of said prototypes, different types of reactant building-blocks representing different roles that said inflows play as participants in said processes, and one or more instances of product building-blocks (808) to represent the outflow(s) from the processes represented by said prototypes;
- said output-input pairs comprise: an output of any of said instances of reservoir building-blocks and an instance of said reactant building-blocks of any of said instances of process building-blocks, and an instance of product building-block and an input of any of said instances of reservoir building-blocks.

28. A method as claimed in Claim 27 wherein said prototype(s) of reservoir building-block(s) (801) are composite comprising each a subworkspace (804) with a set of components that can be displayed to provide visual access to said components, the method being further characterized by:

- said components comprise one or more instances of input building-blocks and output building-blocks, each representing one of said inflow(s) and outflow(s), respectively, to said pool of entities;
- said output-input pairs comprise: an instance of output building-block of any of said instances of reservoir building-blocks and an instance of reactant building-block of any of said instances of process building-blocks, and an instance of product building-block of any of said instances of process building-blocks and an instance of input building-block of any of said instances of reservoir building-blocks.

Patentansprüche

1. Ein Computersystem zum Erzeugen virtueller Modelle von Komplexsystemen, die Prozessormittel, Arbeitsspeichermittel, Speicherungmittel, Inputmittel, Outputmittel, Anzeigemittel und Programmittel umfassen, charakterisiert durch:

- ein oder mehr Datenbanken gespeichert in die genannten Speicherungmittel umfassend Instanzen von Prozeßbausteine (502) und Reservoirbausteine (501) welche Komponenten von ein oder mehr virtuellen Modell von den genannten Komplexsystemen darstellen, jede der genannten Instanzen umfassen eine beliebige Anzahl von Attributen, welche Datentypen enthalten aber nicht beschränkt sind an: Zeichenkette, ganze Zahl oder reelle Zahlen, Logische Werte, Fuzzy Werte, oder Instanzen von Variablen, Parametem, Listen, Arrays, Bildern, oder irgendwelche andere Datenstrukturen, oder Zeigern zu irgendwelchen von den genannten Bauteilinstanzen, Externdateien, URLs, Datenbankaufzeichnungen, oder irgendwelchen anderen Datenstrukturen, im genannten Computersystem oder in einem Netzwerk zugänglich durch das genannte Computersystem, worin:

- jede der genannten Instanzen von Prozeßbausteinen (502) repräsentiert eine von vielen Prozessen von dem genannten Komplexsystem und enthält wenigstens einen Input (503) oder einen Output (505), welcher wenigstens einen Inflow oder einen Outflow von dem genannten Prozeß darstellen, worin die genannte Prozeßbausteine von verschiedenen Typen sein können welche verschieden Typen von Prozessen darstellen;
- jede der genannten Instanzen von Reservoirbausteinen (501) repräsentiert ein Pool von beliebiger Anzahl von Einheiten eines einzelnen Typs oder Zustandes von Entitäten welche verfügbar sind in einem oder mehreren von den genannten Prozessen zu partizipieren und enthält wenigstens einen Input (506) oder einen Output (507), welche wenigstens einen Inflow oder einen Outflow von der genannten Pool darstellen, worin jede der genannten Pools von Entitäten nicht benötigt physikalisch in Reservoiren oder andererseits von anderen Pools von Entitäten im genannten Komplexsystem getrennt zu sein:

- wenigstens ein Input (503) oder ein Output (505) von jeder der genannten Instanzen von Prozeßbausteinen ist verbunden mit einem Output (507) bzw. einem Input (506) von den genannten Instanzen von Reservoirbausteinen, die genannten Verbindungen darstellen daß die genannten Inflows oder Outflows aus der genannten Prozesse sind Outflows bzw. Inflows von den genannten Pools von Entitäten, worin jede der genannten Verbundenen Instanzen von Reservoirbausteinen (501) eine beliebige Anzahl von Input Verbindungen (506) und/oder Output Verbindungen (507) zu irgendwelchen Anzahl von den genannten verbundenen Instanzen von Prozeßbausteinen hat (502), und jede der genannten verbundenen Instanzen von Prozeßbausteinen irgendwelche Anzahl von Input Verbindungen (507) und /oder Output Verbindungen (506) zu einer beliebigen Anzahl der genannten verbundenen Instanzen von Reservoirbausteinen hat;
- die genannte Programmittel haben Zugang zu den genannten gespeicherten Datenbank(en), und umfaßt Mittel zur Integration der genannten alternierenden Instanzen von verbundenen Reservoirbausteinen und Prozeßbausteinen als Knoten von einem oder mehreren mehrdimensionalen Netzwerken von Pathways (Abb. 5), wobei jede der genannten Instanzen in einem oder mehreren von den genannten Pathways enthalten ist, und jeder der genannten Instanzen welche mehrere Input Verbindungen haben sind Knoten wo sich verschiedene Pathways vereinigen, und jede der genannten Instanzen welche mehrere Output Verbindungen haben, sind Knoten wo sich verschiedene Pathways verzweigen, und worin Information und Daten enthalten in den genannten Bausteinen von den Komponenten des genannten Komplexsystems integriert werden und durch die genannte Kette von Verbindungen zu einander in Zusammenhang gebracht werden.

2. Ein Computersystem wie im Anspruch 1 beansprucht weiters ermöglichend quantitative Simulationen von den genannten virtuellen Modellen in dem genannten Computersystem, und charakterisiert durch:

   - mathematische Modelle welche das quantitative Verhalten von den genannten Komplexsystemen charakterisieren sind zerlegt in Netzwerke von quantitativen Strukturen verteilt unter den genannten Bauteilinstanzen (Bildern 12 durch 15) und umfassen State-Variablen, abhängige Variable und Parameter, welche Eigenschaften von den dargestellten Komponenten darstellen, worin die Werte von den genannten Parametern und Initialwerten von den genannten State-Variablen aus repräsentierten Komponenten gemessen, gerechnet oder abgeschätzt werden oder typische Defaultwerte sind;
   - die genannte Programmittel enthalten Simulationsmittel auf den Gesetzen charakteristisch für die genannten Komplexsysteme basiert, um die gegenwartiger Werte von jeder der genannten Variablen über Zeit dynamisch Rechnen zu können durch die Inkorporation der Wert(e) von bestimmten der genannten Parameter und/oder Variable(n) in der Bauteilinstanz von genannter Variable und/oder in verbundenen Bauteileinstanzen enthalten.

3. Ein Computersystem wie im Anspruch 2 beansprucht weiters ermöglichend Gemischte-Typ Simulationen der genannten virtuellen Modelle unter Benutzung quantitativer oder semi-quantitativer Daten, oder heunstischer Information, für verschieden Teile der genannten Komplexsystemen, worin:

   - die genannten Bauteilinstanzen weiters einen zweites Set von ihren korrespondierenden quantitativen Vanablen und/oder Parametern enthalten, wobei ein Set absolut Werte und ein anderes Set normalisierte Werte hat, und
   - das genannte Programmittel enthält weiters Simulationsmittel und/oder Inferenzmittel um der genannten absolut-wertigen Variablen und Parametern von bestimmten Teilen der genannten virtuellen Modellen und der

genannten normalisiert-wertigen Variablen und Parametern von anderen Teilen von den genannten virtuellen Modellen über Zeit dynamisch Rechnen und Integrieren zu können.

4. Ein Computersystem wie im Anspruch 1 beansprucht worin die genannte virtuelle Modelle räumlich unterteilt sind, weiters umfassend Instanzen von Lokalisationsbausteinen von einem oder mehreren Typen (2406, 2408) welche diskret physikalisch oder konzeptuell räumliche Abteilungen charakteristisch für die genannten Komplexsysteme darstellen, um der genannten Instanzen von Prozeßbausteinen (24 10) und Reservoirbausteinen (2414) von den genannten virtuellen Modellen in eine Hierarchie von einer oder mehreren Schichten (2405, 2407, 2409) von den genannten Lokalisationsbausteinen Organisieren zu können.

5. Ein Computersystem wie im Anspruch 1 beansprucht worin die genannte virtuelle Modelle im Zeitabschnitte unterteilt sind, weiters umfassend Instanzen von Zeitbausteine von einem oder mehreren Typen (2421, 2423) welche diskret physikalisch oder konzeptuell Zeitintervalle charakteristisch für die genannten Komplexsysteme darstellen, um der genannten Instanzen von Prozeßbausteinen (2410) und Reservoirbausteinen (2414) von den genannten virtuellen Modellen in eine Hierarchie von einer oder mehreren Schichten (2402, 2422) von den genannten Zeitbausteine Organisieren zu können.

6. Ein Computersystem wie im Anspruch 1 beansprucht weiters umfassend eine oder mehrerer Instanzen von Entitätsbausteinen (1804), wobei jede Instanz eine Beschreibung der Zusammensetzung (1807) von einer repräsentativen Entität von einer Pool dargestellt durch eine Instanz von den genannten Reservoirbausteinen darstellt, wobei die genannten Instanzen von Entitätsbausteinen von korrespondierenden Instanzen der Reservoirbausteine zugänglich sind (1801).

7. Ein Computersystem wie im Anspruch 1 beansprucht, weiters umfassend Programmittel für die dynamisches Bildung interaktiv visueller Darstellungen (3110, 3118, 36) von den genannten mehrdimensionalen Netzwerken von Pathways einschließlich querverbundene visuelle Ansichten von Instanzen von wenigstens einem von der genannten Typen von Bausteinen, worin:

- die Zusammensetzung von Pathways in den genannten Ansichten bestimmt ist durch die genannten Programmittel zum Darstellen entweder der genannten Komplexsysteme oder irgendwelcher ihrer Subsysteme welche vom Benutzer ausgewählte Kriterien erfüllen, wie zum Beispiel: Wählen von beliebigen einzelnen oder mehreren genannten Bauteileinstanzen als Startknoten und Wählen eines aufwärts und/oder abwärts Verlauf aus d. genannten Startknoten; und
- die verbundenen (3114, 3115) visuellen Darstellungen in den genannten Ansichten sind entweder Zeigern zu den genannten gespeicherten verbundenen Bauteileinstanzen oder Darstellungen von Klones von den genannten gespeicherten verbundenen Bauteileinstanzen.

8. Ein Computersystem wie im Anspruch 1 beansprucht, weiters umfassend Programmittel für die Durchführung von Anfragen (Abb. 30) basiert auf die genannten aufwärts oder abwärts aufeinanderfolgenden Verbindungen aus irgendwelcher von den genannten Bauteileinstanzen angewählt als Referenz.

9. Ein Computersystem wie im Anspruch 1 beansprucht worin die genannte Prozeßbausteine (843 837) Kompositionen sind jeder umfassend einen Untertafel (809, 836) mit je einem Set von Komponenten welche angezeigt werden können um den visuellen Zugang zu den genannten Komponenten zu ermoglichen und umfassend:

- eine oder mehr Instanzen von Reactantbausteine (835), welche verschieden Inflows am der Prozeß repräsentiert durch die genannte Instanz von Prozeßbausteinen darstellen, jeder verbunden (834) mit einem Output von einer Instanz von Reservoirbausteinen (801), wobei verschiedener Typen von Reactantbausteinen (820, 825, 830, 835, 838, 839) verschieden Rollen darstellen daß die genannten Inflows als Teilnehmer in den genannten Prozessen spielen;
- eine oder mehrere Instanzen von Produktbausteinen (808), welche den Outflow(s) aus Prozessen repräsentiert durch die genannten Prozeßbausteine darstellen, jeder verbunden zu einem Input von einem Reservoirbaustein (801).

10. Ein Computersystem wie in Anspruch 9 beansprucht worin:

- die genannten Reservoirbausteine (801) Kompositionen sind, jede umfassend einen Untertafel (804) mit einem Set von Komponenten verbunden mit Komponenten von Prozeßbausteinen, welche angezeigt sein kann um den visuellen Zugang zu den genannten Komponenten zu ermöglichen und durch die oben genannten virtuellen Modelle zu navigieren, wobei die genannten Komponenten umfassen:

- eine oder mehrere Instanzen von Inputbausteinen (806), wobei jeder einen von den genannten Inflow(s) zu dem genannten Pools von Entitäten repräsentiert, und jeder mit (807) einer von den genannten Instanzen von Produktbausteinen (808) verbunden ist; und
- eine oder mehrere Instanzen von Outputbausteinen (833), wobei jeder einen von den genannten Outflow(s) aus dem genannten Pools von Entitäten repräsentiert, und jeder mit (834) einer von den genannten Instanzen von Reactantbausteinen (835) verbunden ist;
- Navigationsmittel assoziiert mit den genannten verbundenen Instanzen von Reactantbausteinen Produktbausteinen, Inputbausteinen, und Outputbausteinen ermöglichen das Anzeigen von Komponenten von irgendwelcher Instanzen von Prozeßbausteinen oder Reservoirbausteinen aufwärts und/oder abwärts von irgendwelcher angewahlte Instanz.

11. Eine Methode für die Benutzung virtueller Modelle (Abb 5) von Komplexsystemen um Überwachung und/oder Steuerung Systeme in einem Computersystem (112) zu Entwerfen, umfassend Prozessormittel, Arbeitsspeichermittel, Speicherungsmittel, Inputmittel, Outputmittel, Anzeigemittel, Programmittel (114), und eine gespeicherte Datenbank von modularen Bausteinen, welche Komponenten von ein oder mehreren Komplexsystemen darstellen, wobei die Methode ist weiters charakterisiert durch:

    - Ladung der genannten Datenbank (116) in die genannten Arbeitsspeichermittel, die genannte Datenbank umfassend eine beliebige Anzahl von Instanzen von Prozeßbausteinen (502) und Reservoirbausteinen (501), worin:
    - jede der genannten Instanzen von Prozeßbausteinen (502) einer der Prozessen von dem genannten Komplexsystem repräsentiert, und wenigstens einen Input (503) oder einen Output enthält (505) welche wenigstens einen Inflow oder einen Outflow von dem genannten Prozeß darstellt, wobei verschieden Typen von den genannten Prozeßbausteinen verschieden Typen von den genannten Prozessen darstellen;
    - jede der genannten Instanzen von Reservoirbausteinen (501) repräsentiert ein Pool von beliebiger Anzahl von Einheiten eines einzelnen Typs oder Zustandes von Entitäten welche verfügbar sind in einem oder mehreren von den genannten Prozessen zu partizipieren(502), und enthält wenigstens einen Input (506) oder einen Output (507) welcher wenigstens einen Inflow oder Outflow von der genannte Pool darstellt, wobei jede der genannten Pools von Entitäten nicht benötigt physikalisch in ein Reservoir oder andererseits von anderen Pools von Entitaten im genannten Komplexsystem getrennt zu sein;
    - irgendwelche von den genannten Input(s) (503) oder Output(s) (505) von irgendwelchen von der genannten Instanzen von Prozeßbausteinen ist mit einem Output (507) bzw. Input (506) von eine Instanz von den genannten Reservoirbausteinen (501) verbunden, und irgendwelche von den genannten Input(s) (506) oder Output(s) (507) von irgendwelchen von den genannten Instanzen von Reservoirbausteinen sind mit einer beliebigen Anzahl von Output(s) bzw Input(s) von Instanzen von Prozeßbausteinen verbunden;
    - ein Anzahl von den erwähnten alternierenden Instanzen von den genannten Reservoirbausteinen und Prozeßbausteinen in Serie verbundene Knoten welche einen Pathway bestimmen;
    - Auswahl durch das genannten Programmittel der Set von Bauteileinstanzen gespeichert in genannten Datenbank(en) welche Knoten von Pathways sind die aufgenommen werden sollen in einem virtuellen Modell von der Subsystem der für Überwachung und/oder Steuerung einzusetzen ist berücksichtigend Benutzerauswahl sowie: irgendwelche ein oder mehrere von den genannten Bausteinen als Startknoten, und einer aufwärts und/oder abwärts Richtung von dem genannten Startknoten; und
    - dynamische Integrierung durch die genannten Programmittel des genannten Sets von angewählte Instanzen in ein mehrdimensionalen Netzwerken von Pathways (Abb. 5), wobei jede der genannten Instanzen in ein oder mehreren von den genannten Pathways enthalten ist, und jener genannten Instanzen welche viele Input Verbindungen oder viele Output Verbindungen hat, sind Knoten wo sich verschiedene Pathways vereinigen bzw. verzweigen.

12. Eine Methode wie im Anspruch 11 beansprucht weiter ermöglichend der Erstellung von visuellen Ansichten (Bildern 31, 36) von den genannten mehrdimensionalen Netzwerken von Pathways, weiter umfassend folgende Schritte:

    - Auswahl durch das genannte Programmittel der Set von Instanzen von einem benutzerausgewählten Typ von genannten Bausteinen von genannten virtuellen Modell(en) oder irgendwelchen von ihren Subsystemen, welche andere benutzer-ausgewählte Kriterien trifft, sowie: beliebiger einer oder mehrerer von den genannten Bauteileinstanzen als Startknoten(en) wählen, und einer aufwärts und/oder abwärts Richtung von dem genannten Startknoten wählen, welche in den genannten visuellen Ansichten enthalten sind.

- Erzeugung durch das genannte Programmittel von visuellen Darstellungen (3114) die Zeigern auf, oder alternativ visuelle Darstellungen von Klones von, den genannten gewahlten gespeicherten Bauteileinstanzen sind.
- dynamische Erstellung an der genannten Anzeigemittel eines Layouts (3110, 3118, Bild. 36) mit den genannten visuellen Darstellungen verbunden als Knoten von einem Netzwerk von eines oder mehrere Pathways.

13. Ein Methode wie im Anspruch 11 beansprucht zum Entwerfen von Manipulationstrategien zur Steuerung der Operation von komplexen Biochemischen-Systemen in Anwendungen umfassend Behandlung von Krankheiten, Verbesserung von Viehbestand und Nahrungsmittelproduktion, und die Verbesserung des Environments, unter anderem, wobei die Methode ist weiters charakterisiert durch:

    - Durchsuchung für und Enthüllung durch die genannte Programmittel irgendwelche Anzahl von Instanzen von den genannten Bausteinen der genannten virtuellen Modelle, welche mögliche Ziele für Manipulation repräsentieren, um beliebiger Zielen zu erreichen, sowie das Beeinflussen der Expression von beliebigen Genen, wobei das genannte Set von Instanzen enthalten ist in irgendwelchen Pathway(s) aufwärts von Instanz(en) von Reservoirbausteinen welche Pools von Entitäten darstellen die an der Verwirklichung von den genannten Ziel(en), sowie den genannten Gene(n), von einem Benutzer ausgewählt, beteiligt sind.

14. Ein Methode wie im Anspruch 11 beansprucht zum Entwerfen von Manipulationsstrategien zur Steuerung der Operation von komplexen Biochemischen-Systemen durch Kandidat(en) externe(r) Agent(en), real(en) oder konzipiert(en), in Anwendungen umfassend Behandlung von Krankheiten, Verbesserung von Viehbestand und Nahrungsmittelproduktion, und die Verbesserung des Environments, unter anderem, wobei die Methode ist weiters charakterisiert durch:

    - Expandierung der genannten Datenbank um eine oder mehrere Instanzen von Reservoirbausteine zu umfassen, welche Pools von irgendwelcher Anzahl von Einheiten von den genannte(n) extern(en) Agent(en) darstellen, und eine oder mehrere Instanzen von Prozeßbausteinen, welche Wechselwirkung(en) von den genannten externen Agenten mit ihren Zielen in den genannten komplexen Biochemische-System(en) darstellen;
    - Integrierung durch die genannten Programmittel der genannten neuen Instanzen von Reservoirbausteinen und Prozeßbausteinen, welche die genannten externen Agenten und ihre Wechselwirkungen in dem genannten Netzwerk von Pathways darstellen;
    - Durchsuchen für und Enthüllung durch die genannte Programmittel der Set von Instanzen von den genannten Bausteinen der genannten virtuellen Modelle, welche durch Nutzung irgendwelcher einzelnen oder irgendwelcher Kombinationen von den genannten Pools von externen Agenten beeinflußt sein könnten, wobei das genannte Set von Instanzen enthalten ist in irgendwelchen Pathway(s) abwärts aus den genannten Instanzen von Reservoirbausteinen, welche die genannten externen Agenten darstellen, wie von einem Benutzer ausgewählt.

15. Ein Methode wie im Anspruch 11 beansprucht weiters ermöglichend die Simulation von quantitativem Verhalten von den genannten virtuellen Modellen im genannten Computersystem, worin jede der genannten Bauteileinstanzen weiters eine oder mehrerer quantitative Variable oder Parameter enthalt (Bilder 12 durch 15), welche das quantitative Verhalten bestimmen von Komponenten von den genannten Komplexsystemen repräsentiert durch die genannten Instanzen, und worin die genannte Programmittel weiters Simulationsmittel enthält, wobei die Methode ist weiters charakterisiert durch:

    - Benutzung der genannten Simulationsmittel um dynamisch zu simulieren, in Simuliertezeit oder Echtzeit, die genannten virtuellen Modelle oder ihre Subsysteme durch Rechnen über Zeit die gegenwärtiger Werte von jeder der genannten Variablen durch die Inkorporation der Wert(en) von bestimmten der genannten Parameter (n) und/oder Variable(n) enthalten entweder in der Bauteilinstanz die genannte Variable und/oder in den Bauteileinstanzen verbunden mit der genannten Instanz.

16. Ein Methode wie im Anspruch 15 beansprucht weiters ermöglichend Gemischte-Typ Simulationen der genannten virtuellen Modelle unter Benutzung quantitativer oder semi-quantitativer Daten, oder heuristischer Information, für verschiedene Teile der genannten Komplexsysteme, worin die genannte Bauteileinstanzen weiters ein zweites Set von ihren korrespondierenden quantitativen Variablen und/oder Parametern enthalten, wobei ein Set absolut Werte und ein anderes Set normalisierte Werte hat, wobei die Methode ist weiters charakterisiert durch:

    - Benutzung der genannten Simulationsmittel während ein Simulations lauf um der genannten absolut-wertigen Variablen und Parametern von bestimmten Teile der genannten virtuellen Modelle und der genannten normalisiert-wertige Variablen und Parametern von anderen Teilen von den genannten virtuellen Modellen dynamisch zu rechnen über Zeit und zu integrieren.

17. Eine Methode wie im Anspruch 15 beansprucht weiters ermöglichend die Simulation von quantitativem Verhalten von komplexen Biochemischen-systemen um die Auswirkungen von Manipulationsstrategien in Anwendungen umfassend Behandlung von Krankheiten, Verbesserung von Viehbestand und Nahrungsmittelproduktion, und die Verbesserung des Environments, unter anderem, zu überwachen und bewerten, wobei die Methode ist weiters charakterisiert durch:

   - Anwahl durch das genannten Programmittel der Set von Bauteileinstanzen gespeichert in der genannten Datenbank die aufgenommen werden sollen in einem virtuellen Modell von der Subsystem der simuliert werden sollte um die Realisierung von beliebigen Zielen zu bewerten, sowie die Beeinflussung von dem Expression von beliebigen Genen, wobei das genannte Set von Instanzen enthalten ist in irgendwelchen Pathway(s) aufwärts von Instanz(en) von Reservoirbausteinen welche Pools von Entitäten darstellen die an der Verwirklichung von den genannten Ziel(en), sowie den genannten Gene(n), von einem Benutzer ausgewählt, beteiligt sind.

18. Eine Methode wie im Anspruch 15 beansprucht weiters ermoglicht die Simulation von quantitativem Verhalten von komplexen Biochemischen-systemen um der Auswirkungen von externe(n) Agent(en), real(en) oder konzipiert (en), zu überwachen und bewerten, im Anwendungen umfassend Behandlung von Krankheiten, Verbesserung des Viehbestandes und Nahrungsmittelproduktion, und die Verbesserung des Environments, unter anderem, wobei die Methode ist weiters charakterisiert durch:

   - Expandierung der genannten Datenbank um eine oder mehrere Instanzen von Reservoirbausteinen zu umfassen, welche Pools von irgendwelcher Anzahl von Einheiten von den genannte(n) extern(en) Agent(en) darstellen, und eine oder mehrere Instanzen von Prozeßbausteinen, welche Wechselwirkung(en) von den genannten externen Agenten mit ihren Zielen in den genannten komplexen Biochemischen-systemen darstellen;
   - Integrierung durch die genannten Programmittel der genannten neuen Instanzen von Reservoirbausteinen und Prozeßbausteinen, welche die genannten externen Agenten und ihre Wechselwirkungen in dem genannten Netzwerk von Pathways darstellen; und
   - Anwahl durch das genannten Programmittel der Set von Bauteileinstanzen gespeichert in der genannte die aufgenommen werden sollen in einem virtuellen Modell von der Subsystem der simuliert werden sollte um die Auswirkungen von Nutzung irgendwelcher einzelner oder irgendwelcher Kombinationen von den genannten externen Agenten zu bewerten, wobei das genannte Set von Instanzen enthalten ist in irgendwelchen Pathway (s) abwärts von den genannten Instanzen von Reservoirbausteinen welche die genannte(n) externe(n) Agent (en) darstellen, wie vom Benutzer ausgewählt.

19. Eine Methode wie im Anspruch 15 beansprucht weiters ermöglicht die Synchronisierung von Echtzeitsimulationen von den genannten virtuellen Modellen (116) im genannten Computersystem mit realen Betrieb von dem genannten Komplexsystem (102), durch Integrierung (118) von online-überwachter werte der Variablen (122) von irgendwelchen Komponenten von dem genannten Komplexsystem in der genannten Simulation, weiters umfassen Überwachungsmittel (106, 108) verbunden (120) mit den genannten Inputmittel, wobei die Methode ist weiters charakterisiert durch:

   - Einstellung und Verwendung der genannten Überwachungsmittel um ein oder mehrere Wert(en) von Variable (n) zu überwachen (106), die charakteristisch sind für die genannten Prozesse oder Pools von Entitäten von den genannten Komplexsysteme(n) repräsentiert durch irgendwelche von den genannten Variable(en) (118) enthalten in irgendwelchen von den genannten Instanzen von Reservoirbausteinen (501) oder Prozeßbausteinen (502) von dem genannten virtuellen Modell;
   - Definierung und Benutzung einer oder mehrerer Überwachungschnittstellen (120) im genannten Computersystem zwischen den genannten Überwachungsmittel (106, 108) und den genannten Programmittel (114) um die genannten Überwachungswerte (122) zu beziehen und zu weitergeben an ihre entsprechenden Variablen (118) enthalten im den genannten Bauteileinstanzen;
   - Einstellung jeder der genannten Variablen (118) die der genannten überwachen Werte darstellen, um während einer Simulation ihrer Werte über Zeit dynamisch zu erhalten aus den genannten entsprechend gemessenen Werten, anstatt der genannten simulierten Werte.

20. Eine Methode wie im Anspruch 15 beansprucht weiters ermöglichend die Steuerung von einem Verarbeitungssystem umfassend eines von den genannten realen Komplexsystemen basiert auf eine Echtzeitsimulation von ihrem virtuellen Modell (116), weiters umfassend Steuerungsmittel (138) verbunden mit dem genannten Verarbeitungssystem (102) und den genannten Outputmittel (130), wobei die Methode ist weiters charakterisiert durch:

- Einstellung und Verwendung der genannten Steuerungsmittel (138) um die Operation von beliebigen Komponenten (140) von dem genannten Verarbeitungssystem zu regulieren;
- Definierung und Benutzung den Steuerungscode um Steuerungssignale zu erzeugen zur Weitergabe (132) zu den genannten Steuerungsmittel (138), wobei die genannte Signale ausgelöst während einer Simulation wenn die simulierte werte von bestimmten ein oder mehreren von den genannten Variablen (128) enthalten in genannten Bauteileinstanzen von den genannten virtuellen Modellen bestimmte Schwellenwerten erlangen oder sich innerhalb bestimmter Bereiche befinden;
- Definierung und Benutzung ein oder mehrerer Steuerungschnittstellen (130) in genanntem Computersystem zwischen den genannten Programmittel und der genannten Steuerungsmittel um die genannten Steuerungssignale an die genannten Steuerungsmittel weiterzugeben.

21. Eine Methode zum Bauen virtueller Modelle (Abb.5) von Komplexsystemen in einem Computersystem umfassend Prozessormittel, Arbeitsspeichermittel, Speicherungsmittel. Inputmittel, Outputmittel, Anzeigemittel, Programmittel und eine Bibliothek (2001, 1901) von diversen modularen Prototypen von Bausteinen assoziiert mit Methoden gespeichert in genanntem Computersystem, welches unterschiedlichen Typen von Komponenten von den genannten Komplexsystemen darstellen, wobei die Methode charakterisiert ist durch:

- Erzeugung durch die genannten Programmittel Instanzen von irgendwelchen der genannten Prototypen, aus der genannten Bibliothek vom Benutzer gewählt, zur Darstellung diverser Komponenten irgendwelcher von den genannten Komplexsystemen, worin die genannte Bibliothek umfaßt:
- einen oder mehrere Prototypen von Reservoirbausteinen (1201), welcher Instanzen benützt werden zur Darstellung je einer Pool einer beliebigen Anzahl von Einheiten eines einzelnen Typs oder Zustandes von Entitäten, wobei jede wenigstens einen Input oder einen Output umfaßt, die einen Inflow bzw. Outflow von der genannten Pool darstellen, worin jede der genannten Pools von Entitäten nicht benötigt physikalisch in Reservoiren oder andererseits von anderen Pools von Entitäten im genannten Komplexsystem getrennt zu sein; und
- verschiedene Prototypen von Prozeßbausteinen (2001), von welchen jeder einen von verschieden Typen von Prozessen charakteristisch für die genannten Komplexsysteme darstellt, wobei jede wenigstens einen Input und einen Output umfaßt, welche einen Inflow und einen Outflow von dem genannten Prozeß darstellen;
- Erstellung durch die genannte Programmittel Verbindungen zwischen Output-Input Paare angewählt vom Benutzer, wobei die genannten Output-Input Paare umfassen: einen Output von irgendwelchen der genannten Instanzen von Reservoirbausteinen oder Prozeßbausteinen und einen Input von irgendwelchen der genannten Instanzen von Prozeßbausteinen bzw. Reservoirbausteinen, welche darstellen daß ein Outflow aus der ersten eine Inflow quelle für die zweite Instanz der genannten Bausteine ist;
- Speicherung der genannten Instanzen von den genannten Prototypen und der genannten Verbindungen werden gespeichert im genannten Computersystem; und
- dynamisch Integrierung durch die genannten Programmittel den genannten verbundenen Bausteinen in mehrdimensionalen Netzwerken von Pathways (Abb.5), deren Knoten die genannten alternierend verbundenen Instanzen von Reservoirbausteinen (501) und Prozeßbausteine (502) sind welche Information und Daten der Komponenten repräsentierten durch die genannten verbundenen Instanzen integrieren und in Beziehung bringen, worin jede der genannten Instanzen von Reservoirbausteinen Input Verbindungen von und/oder Output Verbindungen zu irgendwelchen Anzahl der genannten Instanzen von Prozeßbausteinen hat, und worin jede der genannten Instanzen von Prozeßbausteinen Input Verbindungen von und/oder Output Verbindungen zu irgendwelche Anzahl der genannten Reservoirbausteine hat, wobei jede der genannten Instanzen in einem oder mehreren von den genannten Pathways enthalten ist, und jener genannten Instanzen welche viele Input Verbindungen hat ist ein Knoten wo sich verschiedene Pathways vereinigen, und jener genannten Instanzen welche viele Output Verbindungen hat ist ein Knoten wo sich verschiedene Pathways verzweigen.

22. Eine Methode wie im Anspruch 21 beansprucht worin jede der genannten Prototypen von Bausteinen weiters einen oder mehrere quantitative State-Variable oder abhängige Vanable oder Parameter enthält (Bildern 12 durch 15) um quantitative Simulationen von den genannten virtuellen Modellen zu ermöglichen. wobei die Methode ist weiters charakterisiert durch:

- Definierung der Simulationsmittel in dem genannten Computersystem um die gegenwärtiger Werte von jeder der genannten Variablen über Zeit zu rechnen, durch die Inkorporation der Wert(e) von bestimmten genannten Parameter(n) und/oder Variable(n) enthalten entweder in der Bauteilinstanz von einer genannten Variablen und/oder in Bauteileinstanzen verbunden mit die genannte Instanz;
- Einstellung durch die genannten Programmittel Default-Werten der genannten Parametern von den genannten Bauteileinstanzen auf bestimmte Werte definiert in genannten Programmittel für jede der genannten Prototypen, basierten auf typischen Normalwerten charakteristisch für Typen von Komponenten von den genannten Komplexsystemen repräsentiert durch die genannten Prototypen;
- Benutzung der genannten Inputmittel um die genannten Default-Werte durch gemessene gerechnete, oder abgeschatzte Werte von irgendwelchen der genannten Parameter von irgendwelcher der genannten Instanzen für welche die genannten spezifischen Werte verfugbar oder erwunscht sind, zu ersetzen; und
- Einstellung durch die genannten Programmittel Initialwerten für irgendwelche von den genannten State-Variablen basierten auf die Verfügbarkeit von bestimmten anderen spezifischen Daten oder Informationen gespeichert in genannten Bauteilinstanzen, wie im genannten Programm definiert, oder in ihrer Abwesenheit auf Default-Werten im genannten Programm definiert für jede der genannten Prototypen basierten auf den typischen Initialwerten charakteristisch für Typen von Komponenten von die genannte Komplexsystemen repräsentierten durch die genannten Prototypen.

23. Eine Methode wie im Anspruch 22 beansprucht worin der genannte Prototypen von Bausteinen weiters einen zweites Set von ihren korrespondierende quantitativen State-Variablen oder abhangigen Variablen, und/oder Parametern enthalten, wobei ein Set absolut Werte und ein anderes Set normalisierte Werte hat, um dynamisch Gemischte-Typ Simulationen von den genannten Komplexsystemen mit Daten von unterschiedlichen Präzisionsgraden verfugbar fur verschiedene Teile von den genannten Modellen zu ermöglichen, wobei die Methode ist weiters charakterisiert durch:

    - Definierung weitere Simulationsmittel für die dynamisch Integrierung der genannten absolut-wertigen Variablen und Parameter von bestimmten Teilen von den genannten virtuellen Modellen und den genannten normalisiert-wertigen Variablen und Parametern von anderen Teilen von den genannten virtuellen Modellen, unter Benützung von Normalisierungsfunktionen spezifisch definiert für verschieden Typen von Bausteinen.

24. Eine Methode wie im Anspruch 21 beansprucht weiters ermoglichend die Erstellung von virtueller Modelle die räumlich unterteilt sind, worin die genannte Bibliothek weiters eine beliebige Anzahl von Prototypen von Lokalisationsbausteinen enthält (2405, 2407), welche diskret physikalisch oder konzeptuell Raum Abteilungen charakteristisch für die genannten Komplexsysteme darstellen, wobei die Methode ist weiters charakterisiert durch:

    - Erstellung durch die genannte Programmittel Instanzen von den genannten Prototypen von Lokalisationsbausteine vom Benutzer ausgewählt aus der genannten Bibliothek, zum Bauen einer Hierarchie von einer beliebigen Anzahl von Schichten von den genannten Instanzen zum Darstellen der räumlichen Hierarchie charakteristisch für der Komplexsystem das modelliert wird,
    - Organisierung der genannten Instanzen von Prozeßbausteinen (2410) und Reservoirbausteinen (2414) von den genannten virtuellen Modellen um innerhalb der geeigneten Unterschichten Instanzen (2408) von den genannte Lokalisationsbausteine in der genannten Hierarchie enthalten zu sein; und
    - Erstellung durch die genannten Programmittel von Behälterbeziehungen zwischen irgendwelchen von den genannten Instanzen von Lokalisationsbausteinen und anderen Instanzen von Lokalisationsbausteinen, Prozeßbausteinen, und Reservoirbausteinen welche ein oder mehrere Kriterien basierten auf der genannten hierarchischen Organisation erfüllen.

25. Eine Methode wie im Anspruch 21 beansprucht weiters ermöglichend die Erstellung von virtueller Modelle die in Zeitintervalle unterteilt sind, worin die genannte Bibliothek weiters enthält eine beliebige Anzahl von Prototypen von Zeitbausteinen enthält (2404, 2421, 2423), welche diskret physikalische oder konzeptuelle Zeitintervalle charakteristisch für die genannten Komplexsysteme darstellen, wobei die Methode ist weiters charakterisiert durch:

    - Erstellung durch die genannten Programmittel von Instanzen der genannten Prototypen von Zeitbausteinen vom Benutzer ausgewählt aus der genannten Bibliothek, um einer Hierarchie von einer beliebigen Anzahl von Schichten (2402, 2422) von den genannten Instanzen zu Bauen zur Darstellen der Zeitintervall Hierarchie charakteristisch für das Komplexsystem welches modelliert wird;
    - Organisierung der genannten Instanzen von Prozeßbausteinen (2410) und Reservoirbausteinen (2414) von den genannten virtuellen Modellen um innerhalb der geeigneten Unterschichten Instanzen der genannten

Zeitbausteine in die genannte Hierarchie enthalten zu sein; und
- Erstellung durch die genannte Programmittel Behälterbeziehungen zwischen irgendwelchen von den genannten Instanzen von Zeitbausteinen und anderen Instanzen von Zeitbausteinen, Prozeßbausteinen und Reservoirbausteinen, welche ein oder mehrere Kriterien basierten auf die genannte hierarchische Organisation erfüllen.

26. Eine Methode wie im Anspruch 21 beansprucht worin die genannte Bibliothek weiters eine beliebige Anzahl von Prototypen von Entitätsbausteine (Abb. 18) enthält, welche Typen von Entitäten charakteristisch für die genannten Komplexsysteme oder Typen von Komponenten von den genannten Entitäten darstellen, wobei die Methode ist weiters charakterisiert durch:

- Erzeugung durch die genannten Programmittel Instanzen von den genannten Prototypen von Entitätsbausteinen vom Benutzer aus der genannten Bibliothek ausgewählt;
- Einordnung in jede Instanz von Spitzenschicht Entitätsbausteinen (1804), wobei jeder die Beschreibung der Zusammensetzung von einem typisch Entität von irgendwelchen von den genannten Instanzen von Reservoirbausteinen (1801) darstellt, eine Hierarchie von einer beliebigen Anzahl von Schichten (1807, 1809, 1914) von den genannten Instanzen von Entitätsbausteine: die ihre modularen Komponenten darstellen;
- von den entsprechenden Instanzen von Reservoirbausteinen (1801) in welche ein Zeiger zu irgendwelchen von den genannten Instanzen Spitzenschicht angebracht worden ist, die genannten Spitzenschicht Instanzen durch die genannten Programmittel zugänglich (1803) machen.

27. Eine Methode wie im Anspruch 21 beansprucht worin die genannte Prototypen von Prozeßbausteinen (843, 837 Kompositionen sind, wobei jeder einen Untertafel (809, 836) mit je einem Set von Komponenten welche angezeigt werden können um den visuellen Zugang zu den genannten Komponenten zu ermöglichen, wobei die Methode ist weiters charakterisiert durch:

- die genannten Komponenten enthalten eine oder mehrere Instanzen von Reactantbausteinen (835) zur Darstellung verschiedener Inflows charakteristisch für den Typ des Prozesses repräsentiert durch jede der genannten Prototypen, wobei verschiedene Typen von Reactantbausteinen verschiedene Rollen darstellen daß die genannten Inflows als Teilnehmer in den genannten Prozessen spielen, und eine oder mehrere Instanzen von Produktbausteinen (808) zur Darstellung von Outflows aus Prozessen repräsentiert durch die genannten Prototypen;
- das genannte Output-Input Paar umfaßt: einen Output von irgendwelchen der genannten Instanzen von Reservoirbausteinen und eine Instanz von den genannten Reactantbausteinen von irgendwelchen von der genannten Instanzen von Prozeßbausteinen, und eine Instanz von Produktbausteinen und einen Input von irgendwelchen der genannten Instanzen von Reservoirbausteinen.

28. Eine Methode wie in Anspruch 27 beansprucht worin genannte(r) Prototyp(en) von Reservoirbausteinen (801) Kompositionen sind, jeder umfassend einen Untertafel (804) mit je einem Set von Komponenten welche angezeigt werden können um den visuellen Zugang zu den genannten Komponenten zu ermöglichen, wobei die Methode ist weiters charakterisiert durch:

- die genannten Komponenten enthalten eine oder mehrere Instanzen von Inputbausteinen und Outputbausteinen, jeder von welchen einen von den genannten Inflow(s) bzw Outflow(s) für die genannten Pools von Entitäten darstellen;
- die genannten Output-Input Paare umfassen: eine Instanz eines Outputbausteins von irgendwelchen der genannten Instanzen von Reservoirbausteinen und eine Instanz eines Reactantbausteins von irgendwelchen der genannten Instanzen von Prozeßbausteinen, und eine Instanz eines Produktbausteins von irgendwelchen der genannten Instanzen von Prozeßbausteinen und eine Instanz eines Inputbausteins von irgendwelche der genannten Instanzen von Reservoirbausteinen.

Revendications

1. UN système informatique pour générer modèles virtuels de systèmes complexes comprenant moyens du traitement, moyens de mémoire, moyens de stockage, moyens d'entrée, moyens de sortie, moyens d'affichage, et moyens de programmation, caractérisé par

- une ou plus de bases de données enregistrées dans dits moyens de stockage comprenant occurrences de processus-blocs (502) et réservoir-blocs (501), les construction-blocs représentant les composants d'un ou plus de modèles virtuels de dits systèmes complexes chaque de dites occurrences comprenant un nombre illimité d'attributs, quels types de valeurs inclure mais sont pas limité à chaîne de caractères, nombres entiers ou réels, valeurs logiques, valeurs fuzzy, ou occurrences de variables, paramètres, listes, matrices, images, ou autre structure quelconque de données, ou pointeurs à dites occurrences quelconques de construction-blocs, fichiers externes URLs, jeu de données, ou autre structure quelconque de données, dans dit système ordinateur ou dans un réseau accessible par dit système ordinateur, dans lequel:
- chacune de dites occurrences de processus-blocs (502) représente un de nombreux processus de dit système complexe et comporte au moins une entrée (503) ou une sortie (505) représentant au moins un influx ou un outflux de dit processus, et dans lequel dits processus-blocs peut être de types différents représentant les différents types de processus;
- chacune de dites occurrences de réservoir-blocs (501) représente un pool d'un nombre illimité d'unités d'un seul type ou état de entités disponible à participer en une ou plus de dits processus et comporte au moins une entrée (506) ou une sortie (507) représentant au moins un influx ou outflux de dit pool, et dans lequel chacun de dits pools de entités ne nécessite pas d'être physiquement séparé, dans un réservoir ou autrement, de autres pools de entités en dit système complexe;
- au moins une entrée (503) ou une sortie (505) de chaque de dites occurrences de processus-blocs est liée à une sortie (507) ou une entrée (506), respectivement, de dites occurrences de réservoir-blocs, dits liens représentant que dits influxes ou outfluxes de dits processus sont outfluxes ou influxes de dits pools de entités respectivement et dans lequel chacune de dites liée occurrences de réservoir-blocs (501) a un nombre illimité de liens d'entrées (506) et/ou liens de sorties (507) à un nombre illimité de dites liée occurrences de processus-blocs (502), et chacune de dites liée occurrences de processus-blocs a un nombre illimité de liens d'entrées (507) et / ou liens de sorties (506) à un nombre illimité dites liée occurrences de réservoir-blocs;
- dits moyens de programmation ont accès à dite enregistrée base de données(s) et comprennent moyens pour intégrer dites alternant occurrences de réservoir-blocs et processus-blocs liés comme noeuds d'un ou plus de réseaux multidimensionnels (Fig. 5) de chemins, chaque de dites occurrences étant contenue dans une ou plus de dits chemins, chacune de dites occurrences qu'a multiple liens d'entrées est un noeud où différent chemins s'fusionnent, et chacune de dites occurrences qu'a multiple liens de sorties est un noeud où différent chemins branche, et dans lequel l'information et les données contenue dans dits construction-blocs concernant les composants de dit système complexe sont intégrés et rattachés à chaque autre à travers de dites séries de liens.

2. UN système informatique comme revendiqué dans Revendication 1 plus de permettre simulations quantitatives de dits modèles virtuels dans dite système informatique, caractérisé par:
* modèles mathématiques caractérisant le comportement quantitatif de dits systèmes complexes sont décomposé en réseaux de structures quantitatives distribué dans dites occurrences de construction-blocs (Figs 12 à travers de 15) comprenant state variables, variables dépendantes, et paramètres représentant caractéristiques des composants ils représentent, dans lequel valeurs de dits paramètres et valeurs initiales de dites state variables sont mesurées, calculées, ou estimées à partir des composants représenté ou sont représentatif valeurs par défaut;

   - dits moyens de programmation comporte moyens de simulation sur la base de lois caractéristiques de dites systèmes complexes pour dynamiquement calculer les valeurs actuelles de chaque de dites variables dans le temps en incorporant le(s) valeur(s) de certain de dits paramètre(s) et/ou variable(s) contenue dans la même occurrence de construction-bloc de dite variable et/ou occurrences de construction-blocs liée à dite occurrence.

3. UN système informatique comme revendiqué dans Revendication 2 plus de permettre simulations de type mixte de dits modèles virtuels utilisant données quantitatives or semi-quantitatives, ou information heuristique, pour différentes parts de dits systèmes complexes, dans lequel:

   - dites occurrences de construction-blocs comprennent supplémentairement un deuxième série de leur correspondants quantitatif variables et/ou paramètres une série ayant valeurs absolues et autre série ayant valeurs normalisées,
   - dits moyens de programmation comprennent supplémentairement moyens de simulation et/ou moyens d'inférence pour dynamiquement calculer dans le temps et intégrant dites variables et paramètres avec valeurs absolues de certain parts de dits modèles virtuels et dites variables et paramètres avec valeurs normalisées de autres parts de dits modèles virtuels.

4. UN système informatique comme revendiqué dans Revendication 1 dans lequel dits modèles virtuels sont compartimenté dans l'espace, comprenant supplémentairement occurrences d'emplacement-blocs, les construction-blocs d'une ou plus types (2406, 2408) représentant discret physique ou conceptuel espace compartiments caractéristiques de dits systèmes complexes, utilisés pour organiser dites occurrences de processus-blocs (2410) et réservoir-blocs (2414) de dits modèles virtuels en une hiérarchie d'une ou plus de couches (2405, 2407, 2409) de dits emplacement-blocs.

5. UN système informatique comme revendiqué dans Revendication 1 dans lequel dits modèles virtuels sont compartimenté en intervalles de temps, comprenant supplémentairement occurrences de temps-blocs, les construction-blocs d'une ou plus types (2421 2423) représentant discret physique ou conceptuel intervalles de temps caractéristiques de dits systèmes complexes, utilisés pour organiser dites occurrences de processus-blocs (2410) et réservoir-blocs (2414) de dits modèles virtuels en une hiérarchie d'une ou plus de couches (2402, 2422) de dits temps-blocs.

6. UN système informatique comme revendiqué dans Revendication 1 comprenant supplémentairement une ou plus d'occurrences de entité-blocs (1804) chaque occurrence représentant la description de la composition (1807) d'une entité représentative d'un pool représenté par une occurrence de dits réservoir-blocs, dites occurrences de entité-blocs étant accessible à partir des occurrences correspondantes de réservoir-blocs (1801).

7. UN système informatique comme revendiqué dans Revendication 1, comprenant supplémentairement moyens de programmation pour dynamiquement générer interactive visualisations (3110, 3118, 36) de dits réseaux multidimensionnels de chemins comprenant représentations visuels interconnectés d'occurrences de au moins un de dits types de construction-blocs, dans lequel:

- la composition de chemins en dites visualisations est déterminée par dits moyens de programmation pour représenter dits systèmes complexes ou leur subsystèmes quelconques que répondent à critères sélecté par l'utilisateur, telle que: sélectionner un ou plus de dites occurrences quelconques de construction-blocs comme le(s) noeud(s) de départ et sélectionner un direction en amont et/ou en aval à partir de dit(s) noeud(s); et
- las représentations visuels connectés (3114, 3115) en dites visualisations sont pointeurs à dites enregistrée liée occurrences de construction-blocs ou représentations de clones de dites enregistrée liée occurrences de construction-blocs.

8. UN système informatique comme revendiqué dans Revendication 1 comprenant supplémentairement moyens de programmation pour réaliser requêtes (Fig. 30) sur la base de dits liens successifs en amont ou en aval à partir de dites occurrences quelconques de construction-blocs sélectionné comme référence.

9. UN système informatique comme revendiqué dans Revendication 1 dans lequel dits processus-blocs (843, 837) sont composite, chaque comprenant un subpanneau (809, 836) avec une série de composants que peuvent être affiché pour fournir accès visuel à dits composants, comprenant:

- une ou plus d'occurrences de réactant-blocs (835), représentant différent influxes à le processus représenté par dit occurrence de processus-bloc, chacune liée (834) à une sortie d'une occurrence de réservoir-bloc (801), les types différents de réactant-blocs (820, 825, 830, 835, 838, 839) représentant rôles différents que dits influxes jouer comme participants dans dit processus;
- une ou plus d'occurrences de produit-blocs (808) représentant outflux(s) du processus représenté par dit processus-bloc, chacune liée à une entrée d'un réservoir-bloc (801).

10. UN système informatique comme revendiqué dans Revendication 9 dans lequel dits réservoir-blocs (801) sont composite, chacun comprenant un subpanneau (804) avec une série de composants liée à composants de processus-blocs, que peuvent être affiché pour fournir accès visuel à dits composants et pour naviguer partout dits modèles virtuels, dits composants comprenant:

- une ou plus d'occurrences de entrée-blocs (806), chacune représentant un de dits influx(s) à dit pool de entités, chacune liée à (807) un de dites occurrences de produit-blocs (808); et
- une ou plus d'occurrences de sortie-blocs (833), chacune représentant un de dits outflux(s) de dit pool de entités, chacune liée à (834) un de dites occurrences de réactant-blocs (835);
- moyens de navigation associées avec dites liée occurrences de réactant-blocs, produit-blocs, entrée-blocs et sortie-blocs pour permettre la visualisation des composants de occurrences quelconques de processus ou